EDUCACIÓN

COBATAB

Formación Laboral Básica COBATAB

"Educación que genera cambio"

MODELO ACADEMICO DE LA FORMACIÓN LABORAL BÁSICA EN

DESARROLLO DE SOFTWARE

Formación Laboral Básica COBATAB

MODULO No. I
"PROGRAMACIÓN BÁSICA"

Submódulos:

I.- Programación en C++

II.- Programación en Java

3er. semestre **2026-2027A**

Modelo académico Estudiante

DIDÁCTICA PARA EL ESTUDIANTE

Nombre_____

Plantel _____ Grupo _____ Turno _____

COLEGIO DE BACHILLERES DE TABASCO

MTRO. FERNANDO YRYS HERNÁNDEZ
Director General

MTRA. ANA FANNY HERNÁNDEZ MONTERO
Directora Académica

LIC. CELIS DEL CARMEN RANERO LARA
Subdirectora de Planeación Académica

MTRA. NATIVIDAD VERTIZ HERNÁNDEZ
Jefa de Departamento de Capacitación para el Trabajo

FORMACIÓN LABORAL BÁSICA EN **DESARROLLO DE SOFTWARE**
TERCER SEMESTRE

MÓDULO I. PROGRAMACIÓN BÁSICA

SUBMÓDULO I. PROGRAMACIÓN EN C++
SUBMÓDULO II. PROGRAMACIÓN EN JAVA

En la realización del presente material participaron:

Coordinador: Mtra. María del Rosario Hernández Alejandro
Asesor: Mtro. Ángel Alberto Torres Ramírez
Participantes: Mtro. Roberto Alonso Pérez López

EDICIÓN. 2026-2027A.

Revisado por:
L.C.P. Angela Acosta Díaz

Contenido.

MÓDULO I. PROGRAMACIÓN BÁSICA

GUÍA DE ESTUDIANTES

Estimada comunidad estudiantil del Colegio de Bachilleres de Tabasco: quiero hacer énfasis en nuestra Institución, la cual tiene 5 décadas forjando el aprendizaje de cada uno de los jóvenes, hoy muchos de los que han pasado por nuestras aulas de clases son profesionistas exitosos.

Dentro del marco del 50 aniversario COBATAB, me permito como Director General enviarles un cordial saludo y dirigirme a ustedes a través de esta guía de estudio diseñada para acompañarlos como una herramienta en su formación académica y personal.

En esta etapa de su vida, el aprendizaje autónomo es clave para descubrir sus talentos, fortalecer sus habilidades y construir un futuro con propósito. Cada actividad, cada lectura y cada reto que encuentren en esta guía de estudio está pensado para impulsar su Desarrollo integral.

Les invito a asumir con responsabilidad su proceso de aprendizaje, a colaborar con sus compañeros, docentes y a mantener siempre una actitud de respeto y apertura. En el Colegio de Bachilleres de Tabasco promovemos la equidad y creemos firmemente en el potencial de cada uno de ustedes.

Esta guía de estudio busca ser clara y accesible, con objetivos definidos y contenidos relevantes para su contexto. Les animo a aprovecharla al máximo, a preguntar, reflexionar y aplicar lo aprendido en su vida diaria.

Confío en su capacidad para superar desafíos, alcanzar metas y contribuir positivamente a su comunidad. Cuentan con el apoyo de sus docentes y de todos los que integramos esta institución.

Con entusiasmo y compromiso, sigamos construyendo juntos una educación de calidad, que genera cambio.

Con aprecio y confianza,

Fernando Yrys Hernández
Director General
Colegio de Bachilleres de Tabasco

Presentación

Desde su creación, el Colegio de Bachilleres de Tabasco tiene como objetivo impartir educación de nivel medio superior en la modalidad de bachillerato general, operando desde el año 2009 un plan de estudios bajo un enfoque por competencias a través del Marco Curricular Común definiendo así los rasgos del perfil del egresado de la EMS en donde las competencias genéricas en su conjunto delinean el perfil deseable del joven bachiller.

En este sentido, los planteamientos del Marco Curricular Común de la Educación Media Superior (MCCEMS)[1] buscan una formación integral en el estudiantado mediante el desarrollo de la capacidad creadora, productiva, libre y digna del ser humano, con amor al país, a su cultura e historia. Por ello, el Bachillerato General plantea las diversas Unidades de Aprendizaje Curricular (UAC) y Formaciones para el Trabajo para que con sus estudiantes egresados y egresadas contribuya al logro de su objetivo específico, el cual radica en la "conformación de una ciudadanía reflexiva, con capacidad de formular y asumir responsabilidades de manera comunitaria, interactuar en contextos plurales y propositivos, trazarse metas y aprender de manera continua y colaborativa".

El Colegio de Bachilleres de Tabasco a través de su Competente de Formación Laboral Básica ofrece en ocho de sus planteles la capacitación de **Desarrollo de Software** con la cual se busca desarrollar en el estudiantado competencias laborales básicas, que les permitan aplicar en forma integrada los conocimientos, destrezas, habilidades, actitudes y valores con responsabilidad y autonomía para desenvolverse en contextos específicos del desarrollo personal, académico, social y profesional en situaciones de la vida común, de estudio o trabajo a lo largo de la vida.

En ese contexto, conforme a las necesidades sociales, económicas y demográficas en relación y congruencia con los objetivos y líneas de acción del programa institucional para el Colegio de Bachilleres de Tabasco, se presenta la formación laboral básica en **Desarrollo de Software**, especifica del bachillerato general, con objetivos delimitados acorde a las características del subsistema y la necesidad del conglomerado social en el cual será aplicado, situación que deviene de tendencias académicas que obligan a este subsistema a tener en consideración las áreas de

[1] El cual puede ser consultado a través del siguiente enlace: https://educacionmediasuperior.sep.gob.mx/work/models/sems/Resource/13516/1/image s/Documento%20base%20MCCEMS.pdf

conocimiento y perfil profesional y su estrecha relación con la demanda ocupacional y la oferta laboral, esto con plena convicción de que los estudiantes del Colegio de Bachilleres de Tabasco concluyan satisfactoriamente con sus estudios de educación media superior e ingresen a sus estudios de nivel superior con herramientas y conocimientos que son de interés general conforme al área geográfica y entorno social en el que se desarrollen; por lo que el presente documento se encuentra conformado en apartados mediante los cuales describe la justificación y los elementos claves para su implementación en el aula.

Programa de Estudios de la Formación Laboral Básica en Desarrollo de Software

Semestre	Tercero, Cuarto, Quinto y Sexto	
Créditos	56 totales / 14 por semestre	
Componente	De formación laboral	
Nivel de formación laboral	Básica	
Tiempo asignado	**Mediación docente**	**Estudio independiente**
	448 h. totales 112 h. por semestre	112 h. totales 28 h. por semestre
Sector productivo	Servicios profesionales, Científicos y Técnicos.[2]	

[2] Acorde al RENEC – Registro Nacional de Estándares de Competencia por Sector Productivo. Disponible en https://conocer.gob.nx/renec-registro-nacional-de-estandares-de-competencia-porsector-productivo/

Fundamentación.

La Dirección General del Colegio de Bachilleres de Tabasco en coordinación con la Dirección General del Bachillerato (DGB), acorde a la Nueva Escuela Mexicana (NEM) y al Marco Curricular Común de la Educación Media Superior (MCCEMS), y en su responsabilidad de enfrentar tanto los nuevos retos como los objetivos que de estos se desprenden, actualiza el presente Programa de estudios, el cual responde a una visión de educación integral, pertinente, de calidad y excelencia.

Dicho programa forma parte del Componente de Formación Laboral Básica del Bachillerato General, el cual busca ser un espacio vinculado con el sector productivo, permitiendo al estudiantado no solo cumplir con su trayecto educativo, sino construir su proyecto de vida, con mayores posibilidades de inserción en el mercado de trabajo.

Esto atendiendo al mandato constitucional que en materia educativa, con base en la Reforma Constitucional a los artículos 3°, 31° y 73° de la Constitución Política de los Estados Unidos Mexicanos y de la emisión de la Legislación Secundaria, publicadas el 30 de septiembre de 2019 en el Diario Oficial de la Federación, determinan la reorientación del Sistema Educativo Nacional para "garantizar el derecho a la educación con un enfoque de derechos humanos y de igualdad sustantiva, para incidir en la cultura educativa mediante la corresponsabilidad y el impulso de transformaciones sociales dentro de la escuela y en la comunidad".

Bajo este contexto, es que el Componente de Formación Laboral Básica, adquiere mayor relevancia, pues tendrá como ejes rectores:

- **Enfoque en competencias**, que busca desarrollar las capacidades y habilidades necesarias para el desempeño laboral.
- **Enfoque humanista**, que valora y respeta la diversidad y la dignidad del estudiantado, su potencial creativo, su participación, su bienestar integral y su compromiso social.

Estos enfoques se orientan a promover una educación inclusiva, equitativa y de calidad, que favorezca el aprendizaje permanente y el máximo logro de los aprendizajes.

Así pues, el Componente de Formación Laboral Básica, busca desarrollar en el estudiantado competencias laborales básicas, que les permitan aplicar en forma integrada los conocimientos, destrezas, habilidades, actitudes y valores con responsabilidad y autonomía para desenvolverse en contextos específicos del desarrollo personal, académico, social y profesional en situaciones de la vida común, de estudio o trabajo a lo largo de la vida.[3]

Para lograr dicho propósito, lo que a continuación se presenta es una serie de competencias laborales básicas las cuales permitirán al personal docente el abordaje de aprendizajes con la amplitud y profundidad acorde a los diversos contextos.

Es decir, a partir de estas competencias , se diseñarán estrategias de enseñanza-aprendizaje que permita a las y los estudiantes ser capaces de conducir su vida hacia su futuro con bienestar y satisfacción, con sentido de pertenencia social, conscientes de los problemas sociales, económicos y políticos que aquejan al país, pero también de su entorno inmediato, dispuestos a participar de manera responsable y decidida en los procesos de democracia participativa y a comprometerse en las soluciones de las problemáticas que los aquejan y que tengan la capacidad de aprender a aprender en el trayecto de su vida. [4]

Para lo cual, es de primordial importancia visualizar que debe existir una articulación contextualizada del Componente Laboral Básico, con el Currículum Fundamental y el Ampliado, para garantizar así la transferencia de los conocimientos, experiencias, habilidades, capacidades, actitudes y valores.

Es decir, esta articulación permitirá:

- La transversalidad del conocimiento adquirido en el Currículum Fundamental con las competencias laborales básicas requeridas en el mercado laboral.
- Fomentar el aprendizaje en contextos diversos, utilizando métodos y estrategias de aprendizaje.
- Centrar las necesidades del mercado laboral.

[3] Subsecretaría de Educación Media Superior. (2023b). El currículum laboral en la educación media superior. SEP.
[4] Subsecretaría de Educación Media Superior. (2023b). El currículum laboral en la educación media superior. SEP.

- Fomentar la interacción entre la vida educativa y la comunidad, a fin de propiciar la adquisición de conocimientos y competencias, de acuerdo con el desarrollo biopsicosociocultural del estudiantado.
- Aprovechar, mediante el Programa Aula, Escuela, Comunidad (PAEC), todas las oportunidades para poner en práctica lo que se ha aprendido; su aplicación no se limitará solo a los recursos sociocognitivos y a las áreas de conocimiento, sino que abarcará los aspectos funcionales (competencias laborales) y recursos socioemocionales.
- Mejorar la relación entre la escuela y los sectores productivos.[5]

El componente de Formación Laboral Básico aporta al estudiantado elementos que le permiten iniciarse en diversos aspectos del sector productivo, fomentando una actitud positiva hacia el trabajo y en su caso, su integración al mismo. Los módulos que conforman este programa son el resultado del trabajo colegiado con personal docente que imparte esta capacitación en los diferentes subsistemas coordinados por esta Dirección General, quienes brindan su experiencia y conocimientos buscando responder a los diferentes contextos existentes en el país, así como a la formación de una ciudadanía socialmente útil, para que el estudiantado cuente con la opción de iniciar una ruta laboral que le promueva una proyección hacia las diferentes modalidades laborales.

Aunado a ello, en virtud de que la Educación Media Superior debe favorecer la convivencia, el respeto a los derechos humanos y la responsabilidad social, el cuidado de las personas, el entendimiento del entorno, la protección del medio ambiente, la puesta en práctica de habilidades productivas para el desarrollo integral de los seres humanos, la actualización del presente programa de estudios, incluye temas transversales que según Figueroa de Katra (2005), enriquecen la labor formativa de manera tal que conectan y articulan los saberes de los distintos sectores de aprendizaje que dotan de sentido a los conocimientos disciplinares, con los temas y contextos sociales, culturales y éticos presentes en su entorno; buscan mirar toda la experiencia escolar como una oportunidad para que los aprendizajes integren sus dimensiones cognitivas y formativas, favoreciendo de esta forma una educación incluyente y con equidad.

[5] Subsecretaría de Educación Media Superior. (2023b). El currículum laboral en la educación media superior. SEP.

De igual forma, con base en el fortalecimiento de la educación para la vida, se abordan dentro de este programa de estudios los temas transversales, mismos que se clasifican a través de ejes temáticos de los campos Social, Ambiental, Salud y Habilidad Lectora como el componente básico, con la particularidad de que se complementan con características propias de la formación para el trabajo. Dichos temas no son únicos ni pretenden imitar el quehacer educativo en el aula, ya que es necesario tomar en consideración temas propios de cada comunidad, por lo que el personal docente podrá considerar ya sea uno o varios, en función del contexto escolar y de su pertinencia en cada submódulo:

- Eje transversal Emprendimiento: se sugiere retomar temas referentes a la detección de oportunidades y puesta en práctica de acciones que contribuyen a la demostración de actitudes tales como iniciativa, liderazgo, trabajo colaborativo, visión, innovación y creatividad promoviendo la responsabilidad social.
- Eje transversal Vinculación Laboral: se recomienda abordar temas referentes a la realización de acciones que permiten al estudiantado identificar los sitios de inserción laboral o autoempleo.
- Eje transversal Iniciar, Continuar y Concluir sus estudios de nivel superior: se recomienda abordar temas referentes a los mecanismos que permiten al estudiantado reflexionar sobre la importancia de darle continuidad a sus estudios superiores.

Asimismo, otro aspecto importante que promueve el programa de estudios es la interdisciplinariedad entre asignaturas del mismo semestre, en donde diferentes disciplinas se conjuntan para trabajar de forma colaborativa para la obtención de resultados en los Sugerencia de Evidencia de Evaluación de manera integral permitiendo al estudiantado confrontarse a situaciones cotidianas aplicando dichos saberes de forma vinculada.

Por otro lado, en cada submódulo se observa la relación de las competencias genéricas y profesionales básicas, los conocimientos, las habilidades y actitudes que darán como resultado los Sugerencia de Evidencia de Evaluación,

permitiendo llevar de la mano al personal docente con el objetivo de generar un desarrollo progresivo no sólo de los conocimientos, sino también de aspectos actitudinales.

En este sentido, el rol docente dentro del proceso de enseñanza – aprendizaje, tiene un papel fundamental, como lo establece el Acuerdo Secretarial 4474, ya que el profesorado que imparte el componente de formación profesional, es quien facilita el proceso educativo al diseñar actividades significativas que promuevan el desarrollo de las competencias (conocimientos, habilidades y actitudes); propicia un ambiente de aprendizaje que favorece el conocimiento social, la colaboración, la toma responsable de decisiones y al perseverancia a través del desarrollo de habilidades socioemocionales del estudiantado, tales como la confianza, seguridad, autoestima, entre otras, propone estrategias disciplinares y transversales en donde el objetivo no es la formación de técnicos en diferentes actividades productivas, sino la promoción de las diferentes competencias profesionales básicas que permitan a la población estudiantil del Bachillerato General tener alternativas para iniciar una ruta a su integración laboral, favoreciendo el uso de herramientas tecnológicas de la información y la comunicación; así como el diseño de instrumentos de evaluación que atiendan al enfoque por competencias.

Es por ello que la Dirección General del Bachillerato a través del Trabajo colegiado busca promover una mejor formación docente a partir de la creación de redes de gestión escolar, analizar los indicadores del logro académico del estudiantado, generar técnicas exitosas de trabajo en el aula, compartir experiencias de manera asertiva, exponer problemáticas comunes que presenta el estudiantado respetando la diversidad de opiniones y mejorar la práctica pedagógica, donde es responsabilidad del profesorado realizar secuencias didácticas innovadoras a partir del análisis de los programas de estudio, promoviendo el desarrollo de habilidades socioemocionales y el abordaje de temas transversales de manera interdisciplinar; rediseñar las estrategias de evaluación y generar materiales didácticos.

Finalmente, este programa de estudios brinda herramientas disciplinares y pedagógicas al personal docente, quienes deberán, a través de los elementos antes mencionados, potenciar el papel de los educandos como gestores autónomos de su propio aprendizaje, promoviendo la participación creativa de las nuevas generaciones en la economía, en el ámbito laboral, la sociedad y la cultura, reforzar el proceso de formación de la personalidad, construir un espacio valioso para la adopción de valores y el desarrollo de actitudes positivas para la vida.

Justificación de la Formación Laboral Básica en Desarrollo de Software

La Formación Laboral Básica de Desarrollo de Software se ubica dentro de los Recursos Sociocognitivos, dentro de las disciplinas de Comunicación y Matemáticas, las cuales en el MCCEMS tienen en el proceso de aprendizaje, la función de ampliar, potenciar y consolidar el conocimiento de la experiencia; permitiendo aprovechar y ampliar los conocimientos de las Áreas de Conocimiento, asimismo, los Recursos Sociocognitivos contribuyen a desarrollar capacidades, destrezas, habilidades, actitudes y valores en las y los estudiantes, brindando la posibilidad de construir la propia experiencia, para que sepan qué hacer con el conocimiento que tienen, sepan actuar, entendiendo lo que hacen, comprendiendo cómo participar y colaborar, asumiendo la responsabilidad de las acciones realizadas, sus implicaciones y consecuencias, y transformando los contextos locales y comunitarios en pro del bien común.

Con base en ello, a lo largo del Componente de Formación Laboral Básico, se brindará una formación de carácter genérico y transversal que le permita al estudiantado incorporarse al sector productivo con actividades contextualizadas, de autonomía y responsabilidad mínima, fomentando una actitud positiva hacia el trabajo y en su caso, su integración al mismo.

Con base en ello, a lo largo del Componente de Formación Laboral Básico, se brindará una formación de carácter genérico y transversal que le permita al estudiantado incorporarse al sector productivo con actividades contextualizadas, de autonomía y responsabilidad mínima, fomentando una actitud positiva hacia el trabajo y en su caso, su integración al mismo.

De manera específica, a través de la Formación Laboral Básica en Desarrollo de Software, tiene como eje acercar al estudiantado al desarrollo científico - tecnológico en el sector de los servicios profesionales, científicos y técnicos, como en las Tecnologías de la Información. Evaluando estrategias de solución a problemáticas en diversas áreas, utilizando software de programación de alto nivel, clasificando y diseñando el software mediante diversos lenguajes de programación, para mejorar su entorno, demostrando conciencia social y responsabilidad ante dichas problemáticas.

Con base en lo anterior, esta Formación Laboral Básica tiene el **propósito general** de: Evaluar estrategias para el desarrollo de software, de manera creativa e innovadora; identificando, clasificando y diseñando software mediante diversos lenguajes de programación, para mejorar su entorno, demostrando consciencia social y responsabilidad ante las problemáticas de su contexto, fomentando el trabajo colaborativo y propositivo.

El uso de los lenguajes de programación en esta capacitación demuestra el manejo en forma avanzada de dichos lenguajes para la solución de diversas problemáticas de la vida cotidiana y del contexto, con un desarrollo metodológico, creativo y comunicativo en pro del beneficio personal y social.

De esta manera, las personas egresadas de esta Formación Laboral Básica en Desarrollo de Software se pueden integrar a la vida laboral tanto en instituciones públicas como privadas en los siguientes perfiles ocupacionales: en educación, ciberseguridad, visualización y gestión de datos, comercio electrónico, aplicaciones móviles, robótica y en el desarrollo web.

Los módulos que conforman este programa son el resultado del trabajo colegiado con personal docente que imparte esta capacitación en los diferentes subsistemas coordinados por esta Dirección General, quienes brindan su experiencia y conocimientos buscando responder a los diferentes contextos existentes en el país, así como a la formación de una ciudadanía socialmente útil, para que el estudiantado cuente con la opción de iniciar una ruta laboral que le promueva una proyección hacia las diferentes modalidades laborales.

La capacitación de Desarrollo de software busca que el estudiantado desarrolle las competencias profesionales en el desarrollo de sistemas y aplicaciones para Internet y celular, en donde también se desarrollan las competencias genéricas, la interdisciplinaridad y los ejes transversales de vinculación laboral, emprendimiento y la continuación de sus estudios a nivel superior.

La Capacitación de Desarrollo de Software en la formación para el trabajo del estudiantado está basada en las Normas Técnicas de Competencia Laboral (NTCL) del Consejo Nacional de Normalización y Certificación de Competencias Laborales (CONOCER), son una necesidad para cumplir con las exigencias del mundo actual y de los sectores productivos, porque hoy en día se exige tener trabajadores calificados, capaces de desarrollar en todo momento las áreas de la organización en la cual están inmersos, promoviendo los productos o servicios en el entorno

nacional o internacional, proporcionando las herramientas y técnicas que son básicas para los egresados del nivel medio superior, que les va a permitir vencer todas las fronteras e incorporarse al mundo globalizado por medio de la programación, así como de las Tecnologías de la Información y de la Comunicación (TIC'S) y de la utilización de las Tecnologías del Aprendizaje y del Conocimiento (TAC'S).

Además, para consolidar la formación profesional y el compromiso con el medio ambiente, enfrentando los desafíos del mundo actual en materia de producción y consumo responsable; se consideran las metas de los 17 objetivos de la Agenda 2030 de acuerdo con la situación contextualizada de cada comunidad, para hacer más y mejores cosas con menos recursos, incrementando las ganancias netas de bienestar de las actividades económicas mediante la reducción de la utilización de los recursos, la degradación y la contaminación durante todo el ciclo de vida, logrando al mismo tiempo una mejor calidad de vida.[6]

De igual manera, con el desarrollo de las Tecnologías diversas y el desarrollo de habilidades como el pensamiento crítico, la toma de decisiones, la resolución de problemas, la comunicación efectiva y el trabajo colaborativo. La inteligencia artificial (IA) es la base a partir de la cual se imitan los procesos de inteligencia humana mediante la creación y la aplicación de algoritmos creados en un entorno dinámico de computación. Con esto se pretende que los ordenadores piensen y actúen como los humanos. Tomando tres componentes fundamentales:

- Sistemas computacionales.
- Datos y gestión de estos.
- Algoritmos para tomar decisiones (IA avanzados, código).[7]

Con todo esto, la Formación Laboral Básica de Desarrollo de Software pretende atender las demandas de desarrollo en materia de cómputo y servicios a través de tecnologías aplicadas, así como el desarrollo científico y tecnológico para el progreso humano social y ambiental productivo.

[6] La Agenda 2030 y los Objetivos de Desarrollo Sostenible. Una oportunidad para América Latina y el Caribe.
[7] Inteligencia Artificial aplicada a la solución de problemas nacionales. Consuelo Doddoli, Ciencia UNAM-DGDC.

De esta manera, los estudiantes de Desarrollo de Software pueden proporcionar soluciones de software: aplicaciones, programas y sistemas que permitan atender las necesidades sociales y ambientales para bien de todos. Consolidando las competencias laborales en torno a la situación del contexto de los estudiantes, así como del país.

El Desarrollo de Software considera la metodología STEAM, donde los estudiantes de la capacitación pueden desarrollar proyectos que integran las disciplinas de ciencia, tecnología, ingeniería, arte y matemáticas (STEAM). Con la particularidad de la interdisciplinariedad vinculando los saberes y conocimientos en la vinculación con las instituciones de formación profesional universitarias.

El contenido de la capacitación de Desarrollo de software se divide en cuatro módulos, impartidos a partir del tercer semestre con una carga de 7 horas semanales, cada módulo se integra por dos submódulos en los que se busca desarrollar en el estudiantado la creación de programas con características avanzadas, utilizando C++, Java, Visual .NET, así como la creación de aplicaciones móviles y juegos, además de diseño y programación en robótica, esto con el fin de desarrollar software con bases de datos y creación de páginas web comunicando ideas e información en el entorno laboral y escolar.

El Módulo I. Programación básica, el estudiantado revisará sistemas de información mediante software de programación de alto nivel.

El Módulo II. Base de datos y desarrollo web, en este apartado se analizará la creación de bases de datos y sitios web mediante el software de programación adecuado.

El Módulo III. Aplicaciones para web, el estudiantado utilizará software de aplicación y elementos multimedia para crear sitios web.

El Módulo IV. Aplicaciones móviles, en este apartado se construirá soluciones para aplicaciones móviles -y juegos, así como propuestas para robótica básica respetando su arquitectura y anatomía. Todas las competencias mencionadas hacen posible en el egresado tener los conocimientos, técnicas, métodos y lenguajes necesarios en el desarrollo de software para ingresar a estudios superiores y desempeñarse de forma eficiente, además de desarrollar las habilidades y actitudes necesarias para la realización de una actividad productiva socialmente útil como auxiliar en áreas de desarrollo de software en diferentes instituciones públicas o privadas.

La Capacitación de Desarrollo de Software en la formación laboral básica del estudiantado está basada en las Normas Técnicas de Competencia Laboral (NTCL) del Consejo Nacional de Normalización y Certificación de Competencias Laborales (CONOCER), son una necesidad para cumplir con las exigencias del mundo actual y de los sectores productivos, porque hoy en día se exige tener trabajadores calificados, capaces de desarrollar en todo momento las áreas de la organización en la cual están inmersos, promoviendo los productos o servicios en el entorno nacional o internacional, proporcionando las herramientas y técnicas que son básicas para los egresados del nivel medio superior, que les va a permitir vencer todas las fronteras e incorporarse al mundo globalizado por medio de la programación, así como de las Tecnologías de la información y de la comunicación (TIC'S) y de la utilización de las Tecnologías del aprendizaje y del conocimiento (TAC'S).

Así mismo se señalan algunas normas de competencia laboral vigentes, avaladas por el Consejo Nacional de Normalización y Certificación de Competencias Laborales, que podrán servir de guía para el desarrollo de los módulos:

NMX-J-515-ANCE-2021, Equipos De Control Y Distribución-Requisitos Generales De Seguridad.

NOM-003-SCFI-2014 "Productos Eléctricos – especificaciones de seguridad".

UINF0947.01 Operar las herramientas de cómputo en ambiente de red.

EC0835 Ejecución de software con codificación de comandos y datos orientada a objetos.

EC0160 Desarrollo de código de software.

Ubicación de la Formación Laboral

PRIMER SEMESTRE					SEGUNDO SEMESTRE					TERCER SEMESTRE					CUARTO SEMESTRE					QUINTO SEMESTRE					SEXTO SEMESTRE				
UNIDAD DE APRENDIZAJE CURRICULAR	HD	HI	HT	C	UNIDAD DE APRENDIZAJE CURRICULAR	HD	HI	HT	C	UNIDAD DE APRENDIZAJE CURRICULAR	HD	HI	HT	C	UNIDAD DE APRENDIZAJE CURRICULAR	HD	HI	HT	C	UNIDAD DE APRENDIZAJE CURRICULAR	HD	HI	HT	C	UNIDAD DE APRENDIZAJE CURRICULAR	HD	HI	HT	C
LA MATERIA Y SUS INTERACCIONES	4	1	5	8	CONSERVACIÓN DE LA ENERGÍA Y SUS INTERACCIONES CON LA MATERIA	4	1	5	8	ECOSISTEMAS: INTERACCIONES, ENERGÍA Y DINÁMICA	4	1	5	8	REACCIONES QUÍMICAS: CONSERVACIÓN DE LA MATERIA EN LA FORMACIÓN DE NUEVAS SUSTANCIAS	4	1	5	8	LA ENERGÍA EN LOS PROCESOS DE LA VIDA DIARIA	4	1	5	8	ORGANISMOS: ESTRUCTURAS Y PROCESOS. HERENCIA Y EVOLUCIÓN BIOLÓGICA	4	1	5	8
CIENCIAS SOCIALES I	2	0.5	2.5	4	CIENCIAS SOCIALES II	2	0.5	2.5	4						CONCIENCIA HISTÓRICA I. PERSPECTIVAS DEL MÉXICO ANTIGUO. LOS CONTEXTOS GLOBALES	3	0.75	3.75	6	CONCIENCIA HISTÓRICA II. MÉXICO DURANTE EL EXPANSIONISMO CAPITALISTA	3	0.75	3.75	6	CONCIENCIA HISTÓRICA III. LA REALIDAD ACTUAL EN PERSPECTIVA HISTÓRICA	3	0.75	3.75	6
CULTURA DIGITAL I	3	0.75	3.75	6	CULTURA DIGITAL II	2	0.5	2.5	4						* TALLER DE CULTURA DIGITAL	1	0.25	1.25	2										
PENSAMIENTO MATEMÁTICO I	4	1	5	8	PENSAMIENTO MATEMÁTICO II	4	1	5	8	PENSAMIENTO MATEMÁTICO III	4	1	5	8	* TEMAS SELECTOS DE MATEMÁTICAS I	4	1	5	8						* TEMAS SELECTOS DE MATEMÁTICAS II	4	1	5	8
LENGUA Y COMUNICACIÓN I	3	0.75	3.75	6	LENGUA Y COMUNICACIÓN II	3	0.75	3.75	6	LENGUA Y COMUNICACIÓN III	3	0.75	3.75	6	* PENSAMIENTO LITERARIO	3	0.75	3.75	6										
INGLÉS I	3	0.75	3.75	6	INGLÉS II	3	0.75	3.75	6	INGLÉS III	3	0.75	3.75	6	INGLÉS IV	3	0.75	3.75	6	**	3	0.75	3.75	6	**	3	0.75	3.75	6
HUMANIDADES I	4	1	5	8	HUMANIDADES II	4	1	5	8	HUMANIDADES III	5	1.25	6.25	10	* ESPACIO Y SOCIEDAD	3	0.75	3.75	6	**	3	0.75	3.75	6	**	3	0.75	3.75	6
* LABORATORIO DE INVESTIGACIÓN	3	0.75	3.75	6	* TALLER DE CIENCIAS I	4	1	5	8	* TALLER DE CIENCIAS II	3	0.75	3.75	6	CIENCIAS SOCIALES III	2	0.5	2.5	4	**	3	0.75	3.75	6	**	3	0.75	3.75	6
										✓PROGRAMACIÓN EN C++	3	0.75	3.75	6	✓DESARROLLO DE APLICACIONES DE ESCRITORIO CON BASE DE DATOS	3	0.75	3.75	6	**	3	0.75	3.75	6	**	3	0.75	3.75	6
- *CURRICULUM AMPLIADO*	4	1	5	8	- *CURRICULUM AMPLIADO*	4	1	5	8	✓PROGRAMACIÓN EN JAVA	4	1	5	8	✓PROGRAMACIÓN WEB	4	1	5	8	✓APLICACIONES PARA WEB	3	0.75	3.75	6	✓PROGRAMACIÓN MÓVIL Y JUEGOS	3	0.75	3.75	6
										- *CURRICULUM AMPLIADO*	3	0.75	3.75	6	- *CURRICULUM AMPLIADO*	2	0.5	2.5	4	✓HERRAMIENTAS DE DISEÑO PARA WEB	4	1	5	8	✓DISEÑO Y PROGRAMACIÓN EN ROBÓTICA	4	1	5	8
	30	7.5	37.5	60		30	7.5	37.5	60		32	8	40	64		32	8	40	64	- *CURRICULUM AMPLIADO*	4	1	5	8	- *CURRICULUM AMPLIADO*	3	0.75	3.75	6
																					30	7.5	37.5	60		33	8.25	41.25	66

Formación laboral básica en DESARROLLO DE SOFTWARE

- COMPONENTE DE FORMACIÓN FUNDAMENTAL
- COMPONENTE DE FORMACIÓN FUNDAMENTAL EXTENDIDO OBLIGATORIO
- COMPONENTE DE FORMACIÓN FUNDAMENTAL EXTENDIDO
- COMPONENTE DE FORMACIÓN LABORAL BÁSICO
- COMPONENTE DE FORMACIÓN AMPLIADA

HD: Horas con docente
HI: Horas de estudio independiente
HT: Horas totales
C: Créditos

TOTAL DE HORAS CON DOCENTE SEMANA MES: 187
TOTAL DE HORAS SEMANA MES: 233.75
TOTAL DE HORAS: 3740
TOTAL DE CRÉDITOS: 374

Mapa de la Formación Laboral Desarrollo de Software

Módulo I: **Programación Básica**

Submódulo 1	Programación en C++ 48 horas de mediación docente 12 horas de estudio independiente 6 créditos
Submódulo 2	Programación en Java 64 horas de mediación docente 16 horas de estudio independiente 8 créditos

Nota: Elaboración propia.

Módulo II: **Base de Datos y Desarrollo Web**

Submódulo 1	Desarrollo de Aplicaciones de Escritorio con Base de Datos 48 horas de mediación docente 12 horas de estudio independiente 6 créditos
Submódulo 2	Programación Web 64 horas de mediación docente 16 horas de estudio independiente 8 créditos

Nota: Elaboración propia.

Modulo III: **Aplicaciones Web**

Submódulo 1	Aplicaciones para Web 48 horas de mediación docente 12 horas de estudio independiente 6 créditos
Submódulo 2	Herramientas de Diseño para Web 64 horas de mediación docente 16 horas de estudio independiente 8 créditos

Nota: Elaboración propia.

Módulo IV: **Aplicaciones Móviles**

Submódulo 1	Programación Móvil y Juegos 48 horas de mediación docente 12 horas de estudio independiente 6 créditos
Submódulo 2	Diseño y Programación en Robótica 64 horas de mediación docente 16 horas de estudio independiente 8 créditos

Competencias Laborales Básicas

Hacen referencia a la capacidad para aplicar conocimientos, destrezas, habilidades, actitudes y valores en el desarrollo personal, académico, social y profesional en situaciones de la vida común, de estudio o trabajo para que el estudiantado desarrolle la formación fundamental o laboral básica, que les permite desempeñar **funciones laborales de nivel dos** de competencia, aplicando soluciones a problemas simples en contextos conocidos y específicos.

Tienen validez oficial dentro del Sistema Educativo Nacional (SEN), lo cual se expresa con la emisión del documento que acredita su formación.

Estas competencias se caracterizan por:

- **Pertinencia:** Atiende a las necesidades del sector productivo y son valoradas.
- **Relevancia:** Favorece la empleabilidad y emprendimientos productivos, sin disparidades de género, étnicas o exclusión de grupos vulnerables.
- **Coherencia:** Acorde al tipo educativo de media superior.
- **Suficiencia:** Abarca un proceso completo e independiente a las demás funciones que desempeña quien egresa de la formación para el trabajo, técnica o tecnológica en el sector productivo.
- **Autonomía:** Faculta para el análisis y toma de decisiones.
- **Responsabilidad:** Capacidad para asumir compromisos orientados al logro de objetivos y metas laborales.
- **Variedad:** Abarca la ejecución de actividades rutinarias – no rutinarias, predecibles – impredecibles – contextos diversos.
- **Complejidad:** Moviliza recursos cognitivos, procedimentales y actitudinales en diferentes niveles para la ejecución de actividades y funciones.

De igual forma es importante señalar, que, para el caso de la Formación Laboral Básica, como se señaló anteriormente, se logrará en el estudiantado el nivel de competencia dos, el cual se caracteriza por:

- ✓ Realización de actividades programadas
- ✓ Aplicación de habilidades cognitivas y de comunicación para recibir, transmitir y recordar información
- ✓ Utiliza técnicas, materiales, herramientas y equipamiento que no requieren un nivel de especialización para realizar actividades en contextos conocidos, además del uso de tecnologías de la información y comunicación básicas, actuando con ética, con un enfoque de sostenibilidad y responsabilidad sobre su entorno.[8]

1. Descubre sistemas de información mediante la codificación y compilación de instrucciones pertinentes utilizando software de programación, de forma creativa y responsable, en diversos ámbitos.	**CLBDS1.**
2. Predice sistemas de información a través de sus fundamentos, utilizando software de programación que cumpla con los requerimientos de funcionalidad y rendimiento en diferentes contextos, siendo creativo y reflexionando sobre las consecuencias que se deriven de su toma de decisiones.	**CLBDS2.**
3. Estructura bases de datos relacionadas con situaciones sociales de su comunidad utilzando la programación y mostrando un comportamiento metódico, además de organizado, con la finalidad de promover un pensamiento crítico y analítico.	**CLBDS3.**
4. Construye sitios web creativos y funcionales mediante software de diseño web, para transmitir información a gran escala de forma responsable y empática en diversos contextos.	**CLBDS4.**
5. Prueba la funcionalidad de los sitios web, acorde a las necesidades y requerimientos de los usuarios, para el manejo de datos a gran escala, actuando con eficiencia y reflexionando sobre diferentes posturas de conducirse en el contexto.	**CLBDS5.**

[8] Subsecretaría de Educación Media Superior. (2023b). El currículum laboral en la educación media superior. SEP.

6. Explica la creación de sitios web creativos a través de la utilización de software de aplicación y elementos multimedia para la optimización de páginas web, favoreciendo la solución a diversas situaciones de su entorno, actuando de manera consciente y previniendo riesgos.	**CLBDS6.**
7. Recomienda soluciones mendiante software de aplicación, para aplicaciones móviles y juegos, relacionándose con sus semejantes de forma colaborativa, mostrando disposición al trabajo metódico y organizado en diferentes contextos.	**CLBDS7.**
8. Selecciona propuestas de robótica básica, a través de sus fundamentos, arquitectura y anatomía, en cualquier ámbito, afrontando retos y asumiendo la frustración como parte del proceso.	**CLBDS8.**

Así mismo se señalan algunas normas de competencia laboral vigentes, avaladas por el Consejo Nacional de Normalización y Certificación de Competencias Laborales, que podrán servir de guía para el desarrollo de los módulos:

NMX-J-515-ANCE-2021, Equipos De Control Y Distribución-Requisitos Generales De Seguridad.

NOM-003-SCFI-2014 "Productos Eléctricos – especificaciones de seguridad".

Proceso de evaluación bajo el enfoque en competencias

La evaluación por competencias es un proceso que permitirá mediante la obtención de evidencias conocer el dominio de conocimientos, habilidades y actitudes socioafectivas desarrolladas por el estudiantado.

Dicho proceso deberá ser formativo e integral, es decir, permitirá visualizar no solo el saber, saber hacer y saber ser, sino también el bagaje histórico y cultural lo cual permitirá al estudiantado la comprensión de la realidad social y laboral de los sectores y de la comunidad para a partir de ello lograr su intervención y aporte.

Para ello lo que a continuación se presentan son los principios que orientarán el proceso de evaluación:

1. **Validez**: debe existir correlación entre los resultados de la evaluación y los resultados esperados en situaciones laborales reales.
2. **Confiabilidad**: producir resultados consistentes al evaluar en momentos diferentes y en diversos contextos.
3. **Accesibilidad**: facilitar el acceso a cualquier persona que pueda ser capaz de demostrar el desarrollo de la competencia.
4. **Comunicación**: dar a conocer previamente las condiciones en que se va a evaluar, y posteriormente, los resultados mediante la retroalimentación.
5. **Equidad**: evitar cualquier práctica discriminatoria, es decir, el estudiantado será evaluado bajo los mismos criterios e indicadores.
6. **Flexibilidad**: adaptarse a diferentes modalidades y opciones de formación, así como a las características y necesidades del estudiantado.9

Así pues para poder llevar a cabo lo aquí planteado, se propone la utilización de una amplia gama de instrumentos que permitan visualizar tanto el proceso como el resultado final del aprendizaje del estudiantado, entre los que se encuentran: rúbricas, pruebas de ejecución, portafolios de

[9] Subsecretaría de Educación Media Superior. (2023). El currículum laboral en la educación media superior. SEP.

evidencias, diario de campo o bitácora, organizadores gráficos, ensayos, resolución de ejercicios y problemas, exámenes o pruebas tipo saber, exposición, método de casos, proyectos y debates o discusiones dirigidas, entre otros.

De igual forma, es necesario que la evaluación contemple:

- Autoevaluación: cuando el estudiantado valora su desarrollo y la forma en que aprendió.
- Coevaluación: a través de la retroalimentación entre pares, fomentando la cooperación, la colaboración, la empatía y la crítica constructiva.
- Heteroevaluación: emitida por el personal docente en función de los conocimientos, habilidades, actitudes y valores ponderados en los instrumentos de evaluación.

Finalmente, respecto a los pasos para evaluar las competencias laborales básicas, se presenta a continuación una propuesta elaborada por la Subsecretaría de Educación Media Superior (2023b), la cual ilustra la ruta a seguir y constituye una referencia para el personal docente.

Pasos para evaluar competencias laborales

Fuente: Subsecretaría de Educación Media Superior, 2023b.

Evaluación por competencias

Para evaluar por competencias, se debe favorecer el proceso de formación a través de:

- **La Evaluación Diagnóstica:** se realiza antes de algún proceso educativo (curso, secuencia o segmento de enseñanza) para estimar los conocimientos previos del estudiantado, identificar sus capacidades cognitivas con relación al programa de estudios y apoya al personal docente en la toma de decisiones para el trabajo en el aula.

- **La Evaluación Formativa:** se lleva a cabo durante el proceso educativo y permite precisar los avances logrados en el desarrollo de competencias por cada estudiante y advierte las dificultades que encuentra durante el aprendizaje. Tiene por objeto mejorar, corregir o reajustar su avance y se fundamenta, en parte, en la autoevaluación. Implica una reflexión y un diálogo con el estudiantado acerca de los resultados obtenidos y los procesos de aprendizaje y enseñanza que le llevaron a ello; permite estimar la eficacia de las experiencias de aprendizaje para mejorarlas y favorece su autonomía.

- **La Evaluación Sumativa**: se realiza al final de un proceso o ciclo educativo considerando el conjunto de diversas evidencias que surgen de los aprendizajes logrados.

Instrumentos de evaluación

Con el fin de que el estudiantado muestre el saber hacer que subyace en una competencia, los Sugerencia de Evidencia de Evaluación permiten establecer una estrategia de evaluación, por lo tanto, contienen elementos observables que deben ser considerados en la evaluación tales como:

La participación (discurso y comunicación, compromiso, empeño e iniciativa, cooperación).

- Las actividades generativas (trabajo de campo, proyectos, solución de casos y problemas, composición de textos, arte y dramatizaciones).

- Las actividades de análisis (comprensión e integración de conceptos como interpretación, síntesis y clasificación, toma de decisiones, juicio y evaluación, creación e invención y pensamiento crítico e indagación).

Para ello se consideran instrumentos que pueden agruparse principalmente en (Díaz-Barriga, 2014):

- **Rúbricas:** Son guías que describen las características específicas de lo que se pretende evaluar (productos, tareas, proyectos, exposiciones, entre otras) precisando los niveles de rendimiento que permiten evidenciar los aprendizajes logrados de cada estudiante, valorar su ejecución y facilitar la retroalimentación.

- **Portafolios:** permiten mostrar el crecimiento gradual y los aprendizajes logrados con relación al programa de estudios, centrándose en la calidad o nivel de competencia alcanzado y no en una mera colección al azar de trabajos sin relación. Estos establecen criterios y estándares para elaborar diversos instrumentos para la evaluación del aprendizaje ponderando aspectos cualitativos de lo cuantitativo.

Los trabajos que se pueden integrar en un portafolio y que pueden ser evaluados a través de rúbricas son: ensayos, videos, series de problemas resueltos, trabajos artísticos, trabajos colectivos, comentarios a lecturas realizadas, autorreflexiones, reportes de laboratorio, hojas de trabajo, guiones, entre otros, los cuales deben responder a una lógica de planeación o proyecto.

Con base en lo anterior, los programas de estudio de la Dirección General del Bachillerato al incluir elementos que enriquecen la labor formativa tales como la transversalidad, las habilidades socioemocionales y la interdisciplinariedad trabajadas de manera colegiada y permanentemente en el aula, consideran a la evaluación formativa como eje central al promover una reflexión sobre el progreso del desarrollo de competencias del alumnado. Para ello, es necesario que el personal docente brinde un acompañamiento continuo con el propósito de mejorar, corregir o reajustar el logro del desempeño del bachiller sin esperar la conclusión del semestre para presentar una evaluación final.

Rol del estudiante

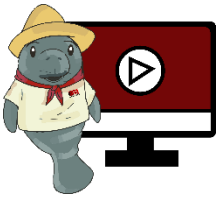
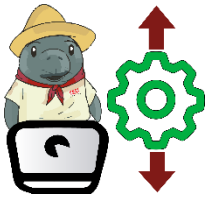
El rol del estudiantado en el proceso educativo no se limita simplemente a recibir información y repetirla, sino que debe ser un agente activo en la construcción de su propio conocimiento y de su identidad. En este sentido, no sólo se trata de aprender a leer y escribir; implica aprender a narrar y comprender su propia vida, tanto como autor o autora de su historia personal, como testigo de su contexto social y cultural. Este proceso es fundamental para que el estudiantado se convierta en un sujeto consciente y crítico de su realidad.

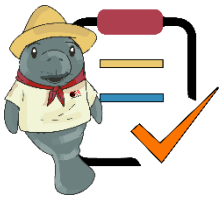
La educación es un motor de transformación social, pero también puede perpetuar las desigualdades existentes al tratar a todos y todas por igual sin considerar la diversidad inherente al estudiantado. La educación debe empoderarles, dándoles las condiciones necesarias para reconocer y cuestionar las desigualdades que les rodean.

Si las y los estudiantes son insertados en una educación que no considera su clase, sexo, género, etnia, lengua, cultura, capacidad, condición migratoria, religión o cualquier otro aspecto de su identidad, es muy probable que se apropien de la idea de que "la escuela no es para ellos y ellas", ya que se enfrentarían constantemente a comentarios o actitudes que les califican de incapaces, ignorantes, indolentes o inútiles terminando por creerlo y asumirlo como verdad. Esta autodesvalorización es una barrera significativa para su desarrollo ya que puede llevar a creer que el conocimiento y la sabiduría pertenecen únicamente a las y los "profesionales" y no reconocen el valor de su propio conocimiento y experiencia.

El rol de las y los estudiantes, entonces, debe ser el de un sujeto activo que desafía y transforma estas narrativas opresivas que fomentan las desigualdades. Debe aprender a valorar su propia voz y experiencia, y a reconocer su capacidad para conocer y transformar su realidad. La educación debe ser un proceso liberador que les permita verse a sí mismos o mismas como agentes de transformación social, capaces de escribir su propia historia y de participar activamente en la construcción de una sociedad más justa y humana.

SIMBOLOGÍA

Icono	Descripción
	**Lectura:** Indica que se realizará la lectura que contiene información de los conocimientos básicos del Programa de Estudio.
	**Material audio visual:** Representa recursos adicionales al contenido de los conocimientos básicos del Programa de Estudio a través de un video mostrado por el docente, indicado por QR.
	**Actividad:** Es momento de realizar una actividad teórica – escrita que contribuye a evidenciar el propósito del aprendizaje esperado de los conocimientos básicos del Programa de Estudio.
	**Práctica:** Hagamos la práctica guiada, que contribuye a la aplicación de los saberes obtenidos con relación al conocimiento básico del Programa de Estudio, se acompaña con el instrumento de evaluación correspondiente.
	**Docente explica:** Se sugiere que esta sección el docente profundice los conocimientos para adecuarlos a su contexto.
	**Actividad SIGA:** Indica el producto que se contempla para la plataforma SIGA.

Icono	Descripción
	**Evaluación Diagnóstica:** Presenta el momento para llevar a cabo la evaluación diagnóstica del submódulo.
	**Instrumento de evaluación:** Documento en el que tanto el estudiante como el docente observan los criterios con los que se evaluará el producto a realizar.
	**Encuadre:** Presenta cada aspecto y el porcentaje de calificación con el que se evaluará el submódulo.
	**Dosificación:** Muestra las fechas en el que los conocimientos básicos del Programa de Estudio se van a desarrollar.
	**PAEC:** Proyecto Escolar Comunitario

Temario

Submódulo 1.
Programación en C++

Unidad 1.- Introducción a la programación y los lenguajes.
- 1.1.- Lenguajes de programación.
- 1.2.- Nivel de abstracción.
- 1.3.- Forma de ejecución.

Unidad 2.- Instalación del editor y compilador de C++.
- 2.1._ Entorno, codificación y compilación.

Unidad 3.- Instrucciones en C++ para aplicaciones básicas.
- 3.1.- Instrucciones de estructura.
  - 3.1.1.- Secuencial.
  - 3.1.2.- Condicional o decisión.
  - 3.1.3.- Cíclicas o repetitivas.

Unidad 4.- Procedimientos y funciones.

Submódulo 2.
Programación en Java

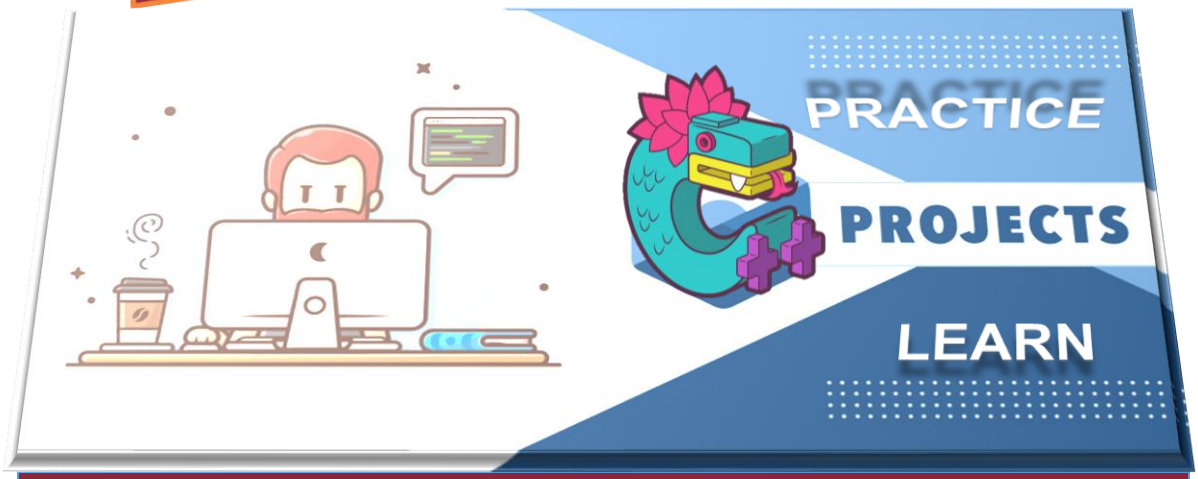
Unidad 1.-Introducción al lenguaje Java.
- 1.1.- Máquina Virtual de Java.
- 1.2.-Kit de desarrollo y entorno de ejecución (JDK).

Unidad 2.- Instalación del IDE de Java.
- 2.1.- Entorno, codificación y compilación.

Unidad 3.- Instrucciones en Java para aplicaciones básicas.
- 3.1.- Instrucciones de estructura.
  - 3.1.1.- Secuencial.
  - 3.1.2.- Condicional o decisión.
  - 3.1.3-Cíclicas o repetitivas.

Unidad 4.- Clases y objetos.

Submódulo 1 Programación en C++



Propósito del Módulo

Utilizar los lenguajes de programación en forma responsable, mediante software de programación de alto nivel, para plantear sistemas de información y desarrollar soluciones en forma sistematizada a diferentes problemáticas, favoreciendo el pensamiento algorítmico y creativo en diferentes ámbitos y contextos.

Sugerencia de Evidencia de Evaluación

- Desarrolla un pensamiento analítico a través del uso de librerías y estructuras de control para la creación de programas en C++ que coadyuven a razonar y ser consciente de situaciones sociales de su vida cotidiana.
- Propone procedimientos y funciones en C++ para el desarrollo de programas, en forma creativa; favoreciendo un comportamiento benéfico para la sociedad.

DOSIFICACIÓN PROGRAMÁTICA

Formación Laboral Básica: DESARROLLO DE SOFTWARE
Módulo I: PROGRAMACIÓN BÁSICA
Submódulo I: Programación en C++ **Clave: C3PC**
Semestre: 3ero **Turno:** Matutino/Vespertino **Periodo:** 2026 – 2027A

Submódulo	Momento	Tiempo (minutos)	Conocimientos	Semana	Fecha inicio	Observaciones
Submódulo I. Programación en C++	Apertura	100 min 100 min 150 min	Bienvenida. Encuadre. Evaluación Diagnóstica. **Unidad 1**.- Introducción a la programación y los lenguajes. 1.1.- Lenguajes de programación. 1.2.- Nivel de abstracción. 1.3.- Forma de ejecución **Unidad 2.-** Instalación del editor y compilador de C++	**1**	31 de Agosto – 04 de Septiembre de 2026	Inicio del semestre 01 de septiembre.
	Desarrollo	350 min	**Unidad 3.-** Instrucciones en C++ para aplicaciones básicas. 3.1.- Instrucciones de estructura.	**2**	07 – 11 de Septiembre de 2026	
		350 min	Instrucciones de estructura: 3.1.1.- Secuencial	**3**	14 – 18 de Septiembre de 2026	16 de septiembre suspensión.
		350 min	Instrucciones de estructura: 3.1.2.- Condicional o decisión.	**4**	21 – 25 de Septiembre de 2026	24 de septiembre jornada de formación y reflexión sobre la gravedad del abuso sexual y el maltrato de las adolescencias.

Submódulo	Momento	Tiempo (minutos)	Conocimientos	Semana	Fecha inicio	Observaciones
Submódulo I. Programación en C++	Desarrollo	350 min	Instrucciones de estructura: 3.1.3.- Repetitiva.	**5**	28 de Septiembre – 02 de Octubre de 2026	Inicia el 22 de septiembre Campaña "Te extrañamos en el salón".
		350 min	**Unidad 4.-** Procedimientos y Funciones.	**6**	05 – 09 de Octubre de 2026	Revisión de portafolios y evaluación sumativa.
	Cierre	350 min	**Unidad 4.-** Procedimientos y Funciones. Situación didáctica: Programando tu salud.	**7**	12 – 16 de Octubre de 2026	

Encuadre de la materia

Criterios de evaluación

Situación didáctica.	Puntaje
Programando tu salud	40%

Actividades	Puntaje
Actividad 01. Cuadro sinóptico de Lenguajes de Programación y compiladores.	2%
Actividad 02. Interfaz gráfica de Dev C++	3%
Actividad 03. Procedimientos y funciones.	5%
Total	10%

Prácticas	Puntaje
Práctica 01. Estructura Secuencial "Promedio de tres calificaciones".	6%
Práctica 02. Estructura Condicional Determinar de tres números dados cual es el mayor y cual el menor.	6%
Práctica 03. Generar los números pares entre 0 y 100.	6%
Práctica 04. Calcular el promedio de tres calificaciones para N estudiantes.	6%
Práctica 05. Dado un número entero N, calcular su factorial (n!).	6%
Total	30%

Examen	Puntaje
Evaluación del submódulo I.	20%
Total	100%

SITUACIÓN DIDÁCTICA

Título:	PROGRAMANDO TU SALUD
Contexto:	De acuerdo con la dirección general de epidemiologia (DGE), Tabasco ocupa el primer lugar en detección de diabetes mellitus tipo 2, cada año se registran casi 3 mil 400 casos nuevos, causada principalmente por los malos hábitos alimenticios e inactividad física. Es imprescindible que los adolescentes conozcan las enfermedades crónicas degenerativas que pueden sufrir, para prevenirlas, y que comprendan que el no tener una vida saludable, puede suponer un riesgo. Por tal motivo se ha encargado a los estudiantes de 3er Semestre de la Capacitación de Desarrollo de Software, que realicen un programa en el lenguaje de programación C++, utilizando las instrucciones, estructuras, procedimientos y/o funciones que mejor convenga para su desarrollo, considerando que a través de la cantidad de glucosa en sangre de una persona en ayunas (detección de diabetes), determine los niveles de glucosa (Normal, Prediabético y diabético), además de mostrar en pantalla, recomendaciones nutricionales por cada nivel de glucosa establecido, por lo cual se debe tomar en cuenta los siguientes criterios: ▪ Si la glucosa está entre 70 y 100 mg/dl se considera una personal SIN DIABETES (Mostrar recomendaciones nutricionales).

	▪ Si la glucosa está entre 100 y 125 mg/dl se considera una personal PRE-DIABETES (Mostrar recomendaciones nutricionales). ▪ Si la glucosa está mayor a 126 mg/dl se considera una personal DIABETES (Mostrar recomendaciones nutricionales).
Propósito	Elaborar un programa creativo en C++, en formato físico o digital, usando el Dev C++ o Editor en C++ sugerido por el docente, proponiendo procedimientos y funciones, que permitan a través de la cantidad de glucosa de una persona (detección de diabetes), determinar los niveles de glucosa (Normal, Prediabético y diabético) y aportar recomendaciones nutricionales por cada nivel de glucosa, en equipos de 5 integrantes para socializarlo ante el grupo.
Conflicto cognitivo	▪ Define que es un programa en C++. ▪ ¿Cuál es el código en C++ para determinar los niveles de glucosa y poder dar recomendaciones? ▪ ¿Qué es la programación estructurada? ▪ ¿Qué es una estructura de control y cuales aplicaste para la resolución de la situación didáctica? ▪ ¿Cómo identificas una condición anidada en la resolución del problema? ▪ ¿Qué recomendaciones darías para adquirir un hábito de alimentación balanceada? 

Evaluación Diagnóstica Submódulo 1

PROGRAMACION EN C++

NOMBRE_____**GRUPO:** _____

INSTRUCCIONES: LEE Y SUBRAYA LA RESPUESTA CORRECTA.

1. ¿Qué es Lenguaje de programación C++?
   a) Un programa para hacer películas
   b) Un lenguaje de secuencias de comandos del lado del cliente
   c) Un lenguaje de programación de alto nivel
   d) Una aplicación para ver vídeos

2. ¿Para que usamos el comando #include?
   a) Para salir
   b) Para incluir las librerías
   c) Para mostrar una vista previa
   d) Para simular un enter

3. Para incluir una biblioteca en la programación de C++ se utiliza la siguiente sentencia:
   a) #include (stdlib)
   b) # include <stdlib>
   c) cin >> variable1;
   d) cout << "biblioteca";

4. ¿Para qué incluimos la librería <iostream>?
   a) Para usar un flujo de entrada y salida
   b) Para ver en negritas el cout
   c) Para incluir las librerías
   d) Para salir

5. En cada programa de C++: Cada variable debe tener su tipo de dato definido y debe haber una función llamada main.
   a) Verdadero
   b) Falso

6. ¿Cuál es el propósito de cin?

   a) Incluye un archivo de encabezado
   b) Imprime un comentario
   c) Imprime el valor de la variable
   d) Toma información (data) del usuario

7. ¿Cuál es el propósito de cout?

   a) Imprime un mensaje en pantalla
   b) Leer un tipo de dato
   c) Es una función
   d) Compila el programa

8. ¿Cuál es el tipo de dato para almacenar enteros?

   a) bool
   b) float
   c) int
   d) char

9. ¿Cuál es el símbolo para desplazarse a una nueva línea (la alternativa para endl)?

   a) \a
   b) \b
   c) \n
   d) \d

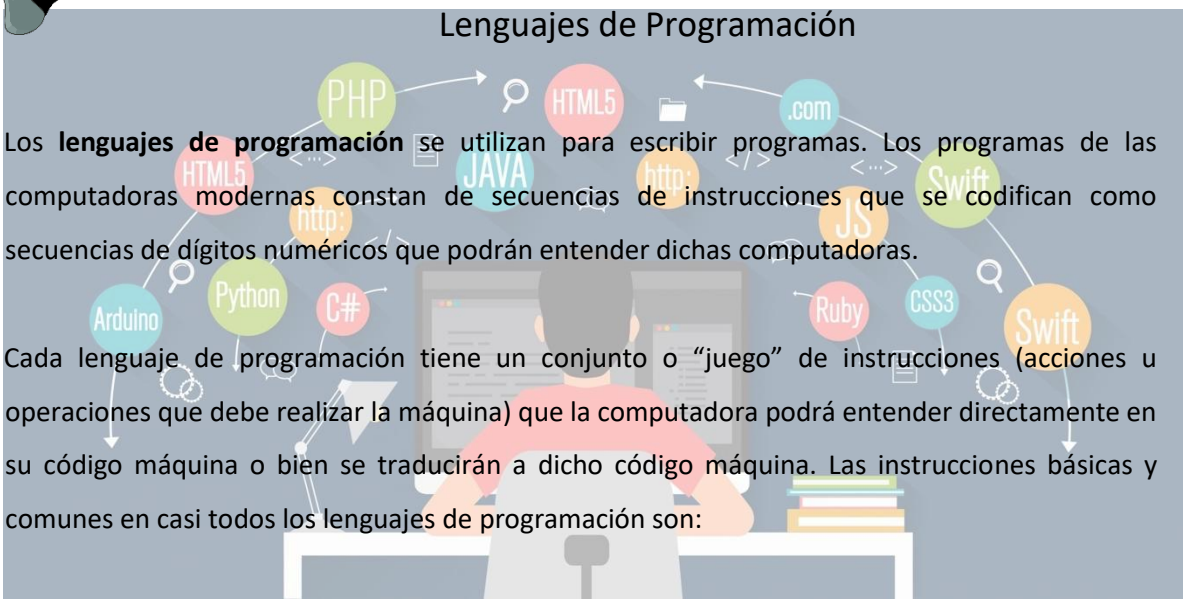
10. En C++ ¿A qué le llamamos función?

   a) Sección del programa que realiza una tarea
   b) Resolución de un problema
   c) Comando de preprocesador
   d) Secuencia de escape

Lectura No. 1 "Introducción a los Lenguajes de Programación"

Instrucciones: *Realiza la Lectura 1. "Introducción a los Lenguajes de Programación" subrayando las ideas principales, y al finalizar realiza la Actividad 1 – Cuadro sinóptico "Lenguajes de Programación"*

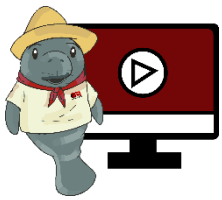
LECTURA 1. "INTRODUCCIÓN A LOS LENGUAJES DE PROGRAMACIÓN"



Los **lenguajes de programación** se utilizan para escribir programas. Los programas de las computadoras modernas constan de secuencias de instrucciones que se codifican como secuencias de dígitos numéricos que podrán entender dichas computadoras.

Cada lenguaje de programación tiene un conjunto o "juego" de instrucciones (acciones u operaciones que debe realizar la máquina) que la computadora podrá entender directamente en su código máquina o bien se traducirán a dicho código máquina. Las instrucciones básicas y comunes en casi todos los lenguajes de programación son:

- **Instrucciones de entrada/salida.** Instrucciones de transferencia de información entre dispositivos periféricos y la memoria central, tales como "leer de..." o bien "escribir en...".
- **Instrucciones de cálculo.** Instrucciones para que la computadora pueda realizar operaciones aritméticas.
- **Instrucciones de control.** Instrucciones que modifican la secuencia de la ejecución del programa.



Además de estas instrucciones y dependiendo del procesador y del lenguaje de programación existirán otras que conformarán el conjunto de instrucciones y junto con las reglas de sintaxis permitirán escribir los programas de las computadoras.

Recursos didácticos

A continuación, se da una clasificación de los lenguajes de programación.

Según el nivel de abstracción

Lenguajes máquina.

El lenguaje máquina es el único que entiende directamente la computadora, ya que está escrito en lenguajes directamente inteligibles por la máquina), utiliza el alfabeto binario, que consta de los dos únicos símbolos **0 y 1**, denominados **bits**. Sus instrucciones son cadenas binarias que especifican una operación y, las posiciones de memoria implicadas en la operación se denominan instrucciones de máquina o código máquina. Fue el primer lenguaje utilizado en la programación de computadoras, pero dejo de utilizarse por su dificultad y complicación, siendo sustituido por otros lenguajes más fáciles de aprender y utilizar.

Lenguajes de bajo nivel (ensambladores).

Son más fáciles de utilizar que los lenguajes máquina, pero al igual que ellos, dependen de la máquina en particular. El lenguaje de bajo nivel por excelencia es el **ensamblador**. El lenguaje ensamblador es el primer intento de sustituir el lenguaje maquina por otro más similar a los utilizados por las personas. A principios de la década de los 50 y con el fin de facilitar la labor de los programadores, se desarrollaron códigos nemotécnicos para las operaciones y direcciones simbólicas. Los códigos nemotécnicos son los símbolos alfabéticos del lenguaje máquina. La computadora sigue utilizando el lenguaje máquina para procesar los datos, pero los programas ensambladores traducen antes los símbolos de código a lenguaje máquina.

Un programa de instrucciones escrito en lenguaje ensamblador por un programador se llama programa fuente. Después de que el ensamblador convierte el programa fuente en código maquina a este se le denomina programa objeto.

Para los programadores es más fácil escribir instrucciones en un lenguaje ensamblador que en código de lenguaje máquina. Los lenguajes de bajo nivel permiten crear programas muy rápidos, pero que son, a menudo, difíciles de aprender. Los lenguajes ensamblador tienen sus aplicaciones muy reducidas, se centran básicamente en aplicaciones de tiempo real, control de procesos y de dispositivos electrónicos.

Lenguajes de alto nivel.

Son los más utilizados como lenguajes de programación. Estos lenguajes permiten que los algoritmos se expresen en un nivel y estilo de escritura fácilmente legible y comprensible por los programadores. Además, los lenguajes de alto nivel suelen tener la característica de "transportabilidad". Es decir, están implementados sobre varias máquinas, de forma que un programa puede ser fácilmente "transportado". El lenguaje de alto nivel permite escribir códigos mediante idiomas que conocemos (español, inglés, etc.) y luego, para ser ejecutados, se traduce al lenguaje de máquina mediante traductores o compiladores.

Ejemplos: ***PASCAL, APL y FORTRAN*** (lenguajes de programación utilizados para aplicaciones científicas), ***COBOL*** (para aplicaciones de procesamiento de datos), ***SNOBOL*** (para aplicaciones de procesamiento de textos), ***LISP y PROLOG*** (para aplicaciones de inteligencia artificial), **C y ADA** (para aplicaciones de programación de sistemas) y ***PL/I*** (para aplicaciones de propósito general).

Programación para la Web

Los programadores pueden utilizar una amplia variedad de lenguajes de programación, incluyendo C y C++ para escribir aplicaciones Web. A continuación, se mencionan algunos lenguajes para el desarrollo de aplicaciones Web:

- **HTML**, técnicamente es un lenguaje de descripción de páginas más que un lenguaje de programación. Es el elemento clave para la programación en la Web.
- **JavaScript,** es un lenguaje interpretado de guionado (scripting) que facilita a los diseñadores de páginas Web añadir guiones a páginas Web y modos para enlazar **esas páginas.**
- **VBScript,** la respuesta de Microsoft a JavaScript basada en VisualBasic.

Lenguajes más populares en 2026.

- **Java,** lenguaje de programación, por excelencia, de la Web.
- **ActiveX,** lenguaje de Microsoft para simular a algunas de las características de Java.
- **C#,** el verdadero competidor de Java y creado por Microsoft.
- **Perl**, lenguaje interpretado de guionado (scripting) idóneo para escritura de texto.
- **XML**, lenguaje de marcación que resuelve todas las limitaciones de HTML y ha sido el creador de una nueva forma de programar la Web. Es el otro gran lenguaje de la Web.
- **AJAX,** es el futuro de la Web. Es una mezcla de JavaScript y XML. Es la espina dorsal de la nueva generación Web 2.0.

Según la Forma de Ejecución

Los procesadores usados en las computadoras son capaces de entender y actuar según lo indican programas escritos en un lenguaje fijo para cada arquitectura, llamado lenguaje de máquina. Todo programa escrito en un lenguaje de alto nivel puede ser ejecutado de dos maneras:

- Lenguajes compilados**:** Antes de poder utilizarse el programa debe utilizarse un traductor llamado "compilador" que se encarga de traducir ("compilar") el programa original ("código fuente") al programa equivalente escrito en lenguaje de máquina o ensamblador ("binario"). Los binarios son los programas ejecutables y los únicos necesarios para el funcionamiento del programa.

- Lenguajes interpretados**:** Cada vez que se usa el programa debe utilizarse un traductor llamado "intérprete" que se encarga de traducir ("interpretar") las instrucciones del programa original ("código fuente") a código máquina según van siendo utilizadas. Para el funcionamiento del programa siempre es necesario disponer del código original y del intérprete.

Bibliografía de lectura 1

Joyanes Aguilar, L. (2010). *Fundamentos de programación en C++.* España: McGraw-Hill

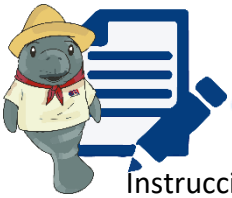
Güimi. (2018). *Lenguajes de programación*. Disponible en: https://guimi.net/descargas/Monograficos/G-Lenguajes_de_programacion.pdf

Yañez Hernández, L. (s/f). Fundamentos de la programación. Madrid: Universidad Complutense. Disponible en: https://www.fdi.ucm.es/profesor/luis/fp/FP01.pdf

Los lenguajes de programación más usados [2022]. (2021, 26 octubre). https://www.crehana.com. https://www.crehana.com/blog/transformacion-digital/lenguajes-de-programacion-mas-usados/

Lenguajes de programación. Disponible en: https://informatica4194.webnode.mx/nosotros/

Códigos Qr de acceso a pagina web y archivos pdf.



Actividad 01.- Cuadro Sinóptico "Lenguajes de Programación"

Instrucciones: En base a la lectura anterior, o con apoyo de otro material bibliográfico, finaliza el siguiente cuadro sinóptico.

LENGUAJES DE PROGRAMACIÓN

LENGUAJE	DESCRIPCIÓN	EJEMPLO
MÁQUINA	Es el único que entiende la computadora digital, en el se pueden utilizar dos símbolos: el cero (0) y el uno (1), este es el lenguaje binario.	Código Máquina. Código ASCII
		Ensamblador
ALTO NIVEL		

INSTRUMENTO DE EVALUACIÓN
LISTA DE COTEJO
Actividad 01. Cuadro sinóptico "lenguajes de programación"

DATOS GENERALES	
Nombre(s) del alumno(s)	**Matricula(s)**
Producto: Cuadro Sinóptico	**Fecha**
Submódulo I: Programación en C++	**Periodo: 2026 – 2027A**
Nombre del docente	**Firma del docente**

CRITERIO	INDICADORES	VALOR OBTENIDO		PONDERACIÓN	CALIF	OBSERVACIONES Y/O SUGERENCIAS DE MEJORA
		SÍ	NO			
1	Entrego en tiempo y Forma.			**2**		
2	Agrego la información correcta.			**2**		
3	Se aprecia el conocimiento obtenido del tema.			**1**		
4	Comprende los datos agregados en el cuadro sinóptico.			**2**		
5	Utiliza la tecnología para desarrollar la actividad.			**2**		
6	Mantienen interés en la comprensión de la actividad.			**1**		
	CALIFICACIÓN					

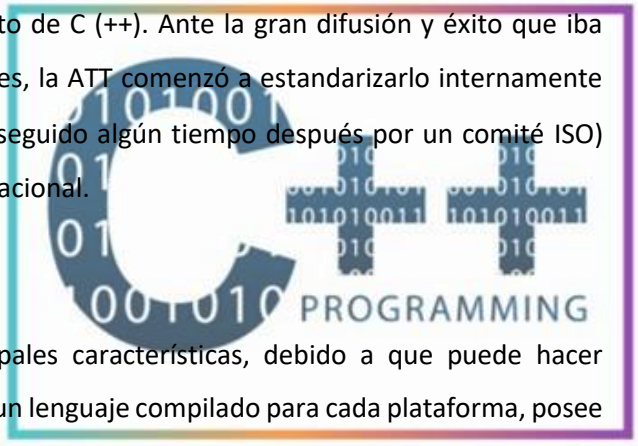
Lectura No. 2 "Programación en C++"

Instrucciones: *Realiza la Lectura 2. "Programación en C++" subrayando las ideas principales, y al finalizar realiza la Actividad 2 – "Interfaz Gráfica de Dev C++"*

LECTURA 2. "PROGRAMACIÓN EN C++"

Lenguaje C++

El lenguaje C++ se comenzó a desarrollar en 1980, siendo su autor ***Bjarne Stroustrup***. Al comienzo era una extensión del lenguaje C que fue denominada C with classes. Este nuevo lenguaje comenzó a ser utilizado fuera de la ATT en 1983. El nombre C++ es también de ese año, y hace referencia al carácter del operador incremento de C (++). Ante la gran difusión y éxito que iba obteniendo en el mundo de los programadores, la ATT comenzó a estandarizarlo internamente en 1987. En 1989 se formó un comité ANSI (seguido algún tiempo después por un comité ISO) para estandarizarlo a nivel americano e internacional.

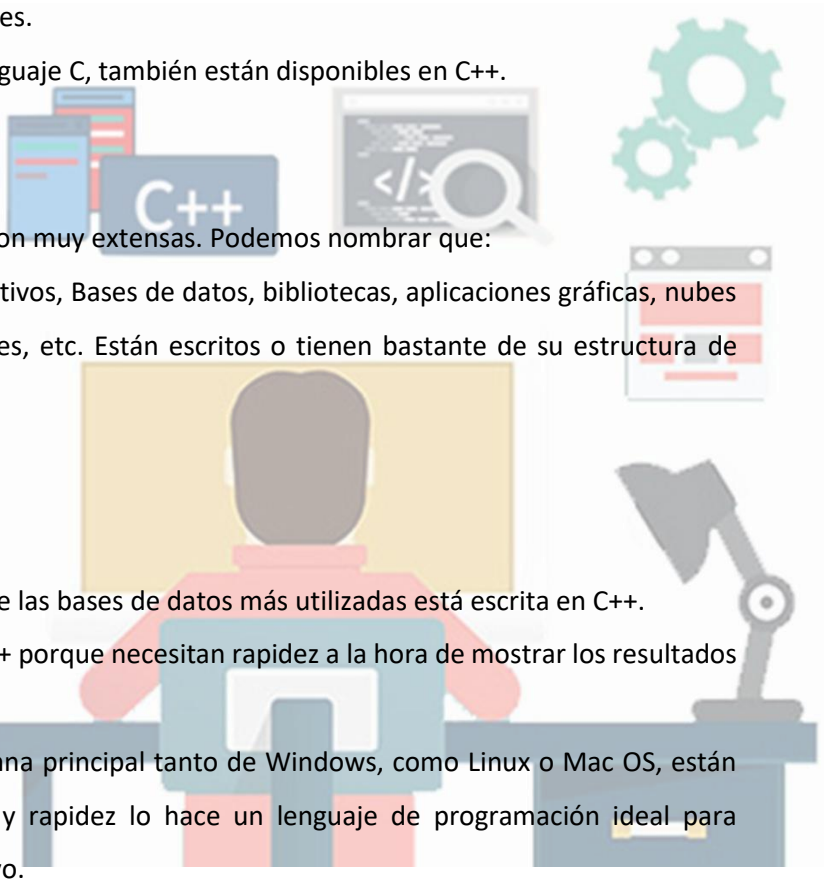


Ventajas de C++

- **Alto rendimiento:** Es una de sus principales características, debido a que puede hacer llamadas directas al sistema operativo, es un lenguaje compilado para cada plataforma, posee gran variedad de parámetros de optimización y se integra de forma directa con el lenguaje ensamblador.
- **Lenguaje actualizado:** A pesar de que ya tiene muchos años, el lenguaje se ha ido actualizando, permitiendo crear, relacionar y operar con datos complejos y ha implementado múltiples patrones de diseño.
- **Multiplataforma:** Permite ser utilizado en diferentes Sistemas Operativos, así como compilarse (Windows, Linux, etc.).
- **Extendido:** Casi cualquier programa o sistema está escrito o tiene alguna parte escrita en estos lenguajes (desde un navegador web hasta el propio sistema operativo).

Características de C++

- **Compatibilidad con bibliotecas:** A través de bibliotecas hay muchas funciones que están disponible y que ayudan a escribir código rápidamente.
- **Orientado a Objetos:** El foco de la programación está en los objetos y la manipulación y configuración de sus distintos parámetros o propiedades.
- **Rapidez:** La compilación y ejecución de un programa en C++ es mucho más rápida que en la mayoría de los lenguajes de programación.
- **Compilación:** En C++ es necesario compilar el código de bajo nivel antes de ejecutarse, algo que no ocurre en otros lenguajes.
- **Punteros:** Los punteros del lenguaje C, también están disponibles en C++.



Aplicaciones y usos de C++

Las aplicaciones del lenguaje C++ son muy extensas. Podemos nombrar que: Navegadores WEB, Sistemas operativos, Bases de datos, bibliotecas, aplicaciones gráficas, nubes (Clouds), videojuegos, compiladores, etc. Están escritos o tienen bastante de su estructura de programación.

Aplicaciones

- **Bases de Datos:** MySQL, una de las bases de datos más utilizadas está escrita en C++.
- **Navegadores WEB:** Utilizan C++ porque necesitan rapidez a la hora de mostrar los resultados en pantalla.
- **Sistemas operativos:** La columna principal tanto de Windows, como Linux o Mac OS, están escritas en C++. Su potencia y rapidez lo hace un lenguaje de programación ideal para programar un sistema operativo.
- **Compiladores:** los compiladores de muchos lenguajes de programación están escritos en C++.
- **Videojuegos:** *C++* es utilizado aún en el mundo de los videojuegos, bien para programar motores gráficos o para alguna parte concreta del videojuego.

Usos

En la programación de máquinas médicas, relojes inteligentes, etc. por su capacidad de estar cerca del lenguaje máquina que otros lenguajes de alto nivel. Por todos estos usos y aplicaciones podemos concluir que la importancia del lenguaje C++ es muy grande y está presente en muchos sitios.

Versiones de C++

La última versión de C++ es la 20 del año 2020 (del año se obtiene el número de versión) y sustituye a la 17 del 2017.

Bibliografía de Lectura 2

Joyanes Aguilar, L. (2010). *Fundamentos de programación en C++.* España: McGraw-Hill

YAÑEZ, L. H. (2014). FUNDAMENTOS DE PROGRAMACION. Madrid, España:
Creative Commons. Pag. 01-568

Robledano, A. (2019). Qué es C++ características y aplicaciones. Disponible en: https://openwebinars.net/blog/que-es-cpp/

Meza González, J. (2020). Curso de C++. Aprende C++ de una buena vez. Disponible en: https://www.programarya.com/Cursos/C++/Estructura

Programación para principiantes. Disponible en:
https://fisiprogramacion.wordpress.com/2014/07/29/c-2-mensajes-por-consola-en-c/

Códigos Qr de acceso a página web y archivos pdf.

"Instalación del editor y compilador Dev C++"

Compilador para PC

Da clic en el link o escanea el código Qr.

Descarga Dev Cpp el cual es el Dev C++ versión 5.5.3.
https://mega.nz/#!cIBDRCTS!83Kg0i_QxXA65Z9bXSO6WlBKKJqGvxVP0nbEfs1VqF4

Instalación Dev C++.
https://www.youtube.com/watch?v=oGeSl58_1Ms

Compilador Web

OnlineGDB beta
online compiler and debugger for c/c++

Compiladores para Móvil

Cxxdroid - C/C++ compiler IDE

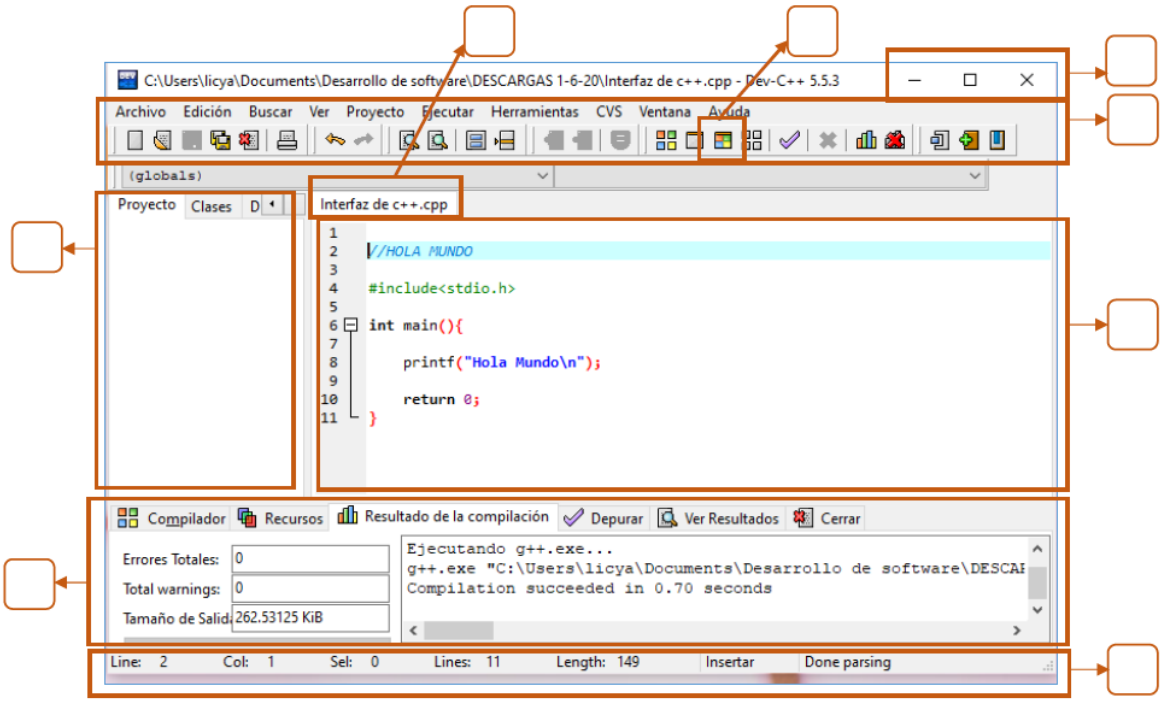
Móvil C [C/C++ Compiler]

C/C++-programming language

C/C++ Program Compiler

Actividad 02 Interfaz Gráfica de Dev C++

Instrucciones: Coloca en el recuadro el número que le corresponda al elemento de la ventana, tomando en cuenta los datos que se presentan en el listado de abajo.



1) Menú y barra de herramientas.
2) Explorador de proyectos y clases e información de depuración.
3) Resultados de la compilación y controles de depuración.
4) Nombre del programa/Código.
5) Icono de compilar y ejecutar.
6) Barra de estado del editor.
7) Área de edición.
8) Botones de control (Minimizar, Maximizar, Cerrar).

INSTRUMENTO DE EVALUACIÓN
LISTA DE COTEJO
Actividad 02. Interfaz Gráfica de C++

DATOS GENERALES	
Nombre(s) del alumno(s)	**Matricula(s)**
Producto: Interfaz Gráfica de Dev C++	**Fecha**
Submódulo I: Programación en C++	**Periodo: 2026 – 2027A**
Nombre del docente	**Firma del docente**

CRITERIO	INDICADORES	VALOR OBTENIDO		PONDERACIÓN	CALIF	OBSERVACIONES Y/O SUGERENCIAS DE MEJORA
		SÍ	NO			
1	Entrego en tiempo y Forma.			**2**		
2	Identifico correctamente la ubicación de los números.			**2**		
3	Se aprecia el conocimiento obtenido de la aplicación.			**1**		
4	Comprende de manera gráfica, el elemento que se señala en la ventana.			**2**		
5	Utiliza la tecnología para desarrollar la actividad.			**2**		
6	Mantienen interés en la comprensión de la actividad.			**1**		
				CALIFICACIÓN		

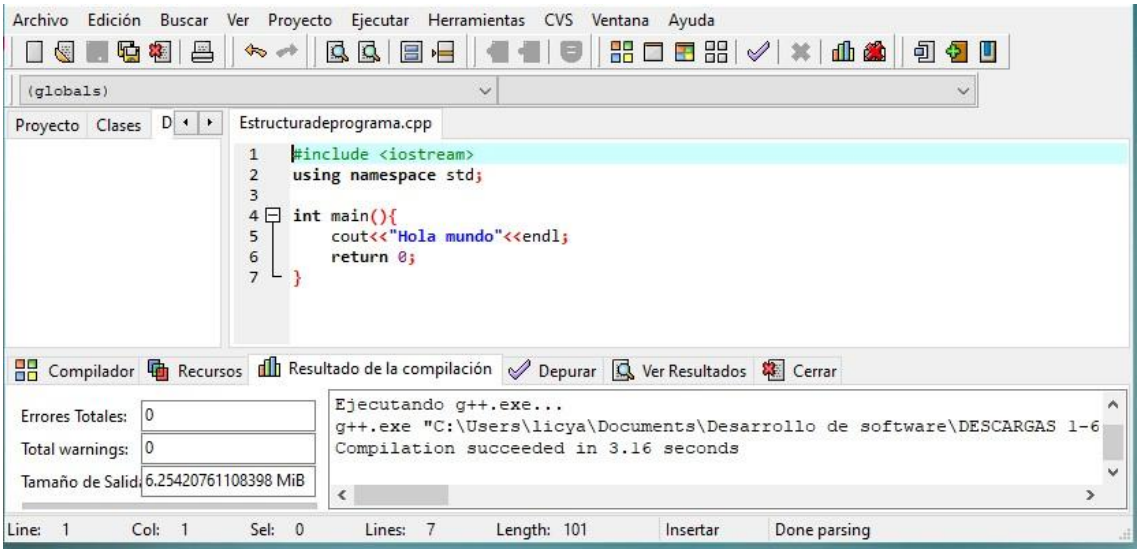
Lectura No. 3 "Sintaxis e Instrucciones Básicas en C++"

Instrucciones: *Realiza la Lectura 3. "Sintaxis e Instrucciones Básicas en C++" subrayando las ideas principales, y al finalizar realiza la práctica guiada – "Saludando ando"*

LECTURA 3. "SINTAXIS E INSTRUCCIONES BÁSICAS EN C++"

Estructura de un programa en C++

Ahora es necesario comprender la estructura de un programa y por ello lo explicaremos a través del código que muestra en pantalla el mensaje "Hola mundo". El código se muestra a continuación y como ves, consta solo de 7 líneas.



- **Línea 1 (#include <iostream>)**: En la primera línea veremos los archivos de cabecera (por eso van a la cabeza), que son las bibliotecas. **Las bibliotecas o librerías son un conjunto de instrucciones predefinidas que nos simplificaran la tarea de la programación.** En este caso estamos incluyendo (**include**) la biblioteca o libreria **iostream**. Vamos a partir un poco la palabra: **<iostream>** i – o – stream. Basa su significado en Input – Output -Stream. Input (entrada), Output (Salida) y Stream (Flujo o transmisión), es decir que la biblioteca o libreria iostream nos servirá para la transmisión de datos por entrada y salida. Para agregar una

biblioteca o librería se empieza con #include seguido por el nombre de la biblioteca o librería entre los signos <> .

- **Línea 2 (using namespace std;):** Esta línea va de la mano cada vez que usemos la biblioteca iostream. Nos permitirá simplificar el código a la hora de leer y mostrar información por consola. Hay una palabra muy especial "std", que en programación veremos varias veces y significa **ST**an**D**ard. **OJO:** Esta línea termina con un punto y coma (;).
- **Línea 3 ("Línea en blanco"):** No hay problema en dejar líneas en blanco en nuestro código, puede que necesitemos hacerlo para acomodarlo o que se vea mejor en algún momento, pero no abuses.

- **Línea 4 (int main() {)**: Main (principal) es la función principal del programa, el tema de funciones lo veremos más adelante pero basta con que sepas que TODOS los programas en C++ tienen su función main, todo lo que esté dentro de las llaves "{" y "}" de tu función main será tu programa. En Code::Blocks, cuando abras una llave "{" inmediatamente se cerrará "}" esto nos ayudará en varias ocasiones, ya que una pesadilla de los programadores que recién empiezan en que a veces ponen el símbolo para cerrar la llave "}" de más o faltan.

- **Línea 5 (cout<<"Hola mundo"<<endl;)**: Esta línea nos permitirá escribir el mensaje *"Hola mundo"* por consola.
  - **cout<<**: cout, o mejor dicho C – Out (Console OUTput o Salida por Consola) nos permitirá escribir mensajes por consola, siempre va de la mano con el operador "<<" llamado ***operador de inserción***.
  - **"Hola mundo":** Es el mensaje que saldrá por consola, solo se mostrará lo que esté entre las comillas dobles.
  - **endl;** : endl (END Line o Fin de Línea) **OJO** la última letra es una L en minúscula, nos servirá para marcar que en ese punto la consola hará un salto de línea. Si no se colocará el endl, en caso de poner más de un solo mensaje, ambos mensajes aparecerán seguidos. **OJO** Toda esta línea termina con punto y coma ";".

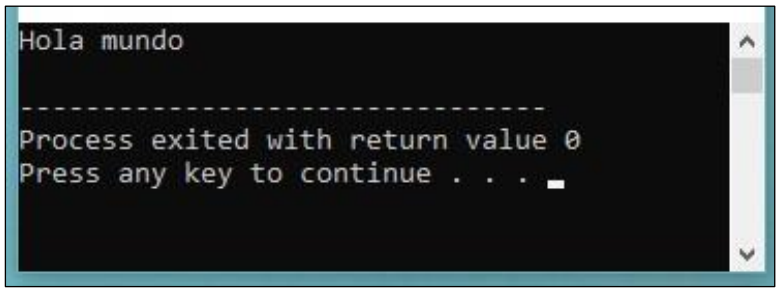
- **Línea 6 (return 0;)** : Return (retornar) nos indica el fin de una función, en este caso la función principal (línea 4) indicaba que era de tipo "int", int significa Integer o Entero, por lo tanto el return debe retornar un entero. El 0(Cero), lo podemos ver como para indicar que nuestro programa terminó con 0 errores. **OJO**: Esta línea termina con punto y coma ";".

- **Linea 7 (})** : La llave para cerrar la función principal, que abrimos en la línea 4. Recuerda que cuando se inserta en automática se abre y se cierra la llave ({ }).

Algo muy importante son los punto y coma (;). El punto y coma marca el fin de una instrucción. Y las llaves ({ }) encierran un bloque de instrucciones. Observa que no todas las líneas necesariamente son instrucciones.

Ahora sí, puedes ejecutarlo presionando F9, para compilar el código y después F10 para compilarlo y ejecutarlo para poder ver el resultado.

Así debería de visualizarse.



Elementos de un programa en C++

A continuación, se describe la definición **léxica** y **sintáctica** de los elementos de un programa: comentarios, identificadores, variables y valores, tipos primitivos, literales, operadores, expresiones y expresiones aritmético – lógicas.

Comentario: Existen 2 tipos de comentarios.

1. Comentario de Bloque. Empieza por /* y termina por */. El compilador(IDE) ignora todo el texto contenido dentro del comentario.

```
/*
*Este programa escribe el texto "Hola Mundo" en la consola
*Utilizando la función printf( );
*/
```

```
1  #include <iostream>
2  using namespace std;
3  //Comentario de 1 linea
4
5  int main(){
6     /*Esto
7       es
8       un
9       comentario
10    */
11    cout<<"Hola mundo"<<endl;
12    return 0;
13 }
```

2. Comentario de línea. Empieza con //. El comentario comienza con estos caracteres y termina al final de la línea.

```
// #include <stdio.h> Las bibliotecas contienen el código objeto de muchos programas que permiten // hacer cosas comunes.
```

El uso de comentarios hace más claro y legible un programa. En los comentarios se debe decir que se hace, para que y cuál es el fin de nuestro programa. Conviene utilizar comentarios siempre que merezca la pena hace una aclaración sobre el programa.

Identificadores: El programador tiene libertad para elegir el nombre de las variables, los métodos y de otros elementos de un programa. Existen reglas muy estrictas sobre los nombres que se utilizan como identificadores de clases, de variables o de métodos.

- Todo identificador debe empezar con una letra que puede estar seguida de más letras o dígitos. Una letra es cualquier símbolo del alfabeto y el carácter '_'. Un dígito es cualquier carácter entre '0' y '9'.
- Los identificadores *Hola, hola, numero, numeroPar, numeroImpar, numero_impar, numero_par, nombre, apellido1* y *apellido2* son válidos. El identificador *1numero* no es válido porque empieza con un dígito, no con una letra.

Cualquier identificador que empiece con una letra seguida de más letras o dígitos es válido siempre que no forme parte de **las palabras reservadas del lenguaje C++**. El lenguaje C++ distingue entre letras mayúsculas y minúsculas, esto significa que los identificadores *numeroPar* y *numeropar* son distintos.

Normas básicas para los identificadores que se deben respetar.

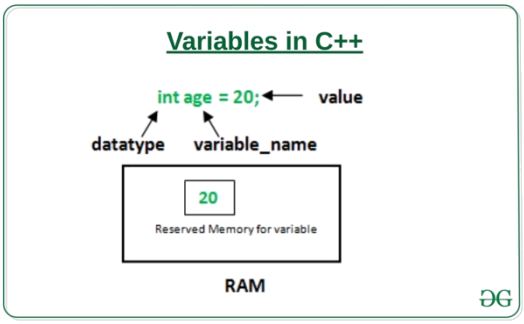
- Los nombres de variables y métodos empiezan con minúsculas. Si se trata de un nombre compuesto cada palabra debe empezar con mayúscula y no se debe utilizar el guión bajo para separar las palabras. Por ejemplo, los identificadores *calcularSueldo*, *setNombre* o *getNombre* son válidos.
- Los nombres de clases empiezan siempre con mayúsculas. En los compuestos, cada palabra comienza con mayúscula y no se debe utilizar el guión bajo para separar las palabras. Por ejemplo, *HolaMundo*, *PerimetroCircunferencia*, Alumno o Profesor son nombres válidos.
- Los nombres de constantes se escriben en mayúsculas. Para nombres compuestos se utiliza el guion bajo para separar las palabras. Por ejemplo, *PI*, *MINIMO*, *MAXIMO* o *TOTAL_ELEMENTOS* son nombres válidos.

Ejemplos de identificadores

Válidos	No válidos	
Letra	1Apellido	//Empieza por numero
letra	Peso Neto	//espacio
variable	Pr€cio	//contiene €
peso_neto	Valor.1	//contiene .
PesoBruto	Tensión	//contiene tilde
Apellido1	Tamaño	//contiene ñ
Apellido_1	int	//palabra reservada

Lenguaje de programación C - David Carmona 2010

Variables y valores: Un programa C++ utiliza variables para almacenar valores, realizar cálculos, modificar los valores almacenados, mostrarlos por la consola, almacenarlos en disco, enviarlos por la red, etc. Una variable almacena un único valor. Se define por un nombre, un tipo y el rango de valores que puede almacenar.



El nombre de una variable permite hacer referencia a ella. Este nombre debe cumplir las reglas aplicables a los identificadores. El tipo indica el formato de los valores que puede almacenar la variable: cadenas de caracteres, valores lógicos, números enteros, números reales o tipos de datos complejos. El rango indica los valores que puede tomar la variable.

Por ejemplo, una variable de tipo número entero, con nombre *mesNacimiento* puede almacenar valores positivos y negativos, lo que no tiene sentido cuando se trata de meses del año. El rango válido para esta variable sería de 1 a 12. Para declarar una variable en C++ se indica el tipo y su nombre.

```
#include<iostream>
using namespace std;
// global variable
int global = 5;

// main function
int main()
{
    // local variable with same
    // name as that of global variable
    int global = 2;

    cout << global << endl;
}
```

Global Variable

Local variable

int mesNacimiento;

En este ejemplo se indica que la variable es de tipo entero (int) y su nombre es *mesNacimiento*. Una vez declarada una variable, se puede utilizar en cualquier parte del programa referenciándola por su nombre. Para almacenar un valor en una variable se utiliza el operador de asignación y a continuación se indica el valor, por ejemplo 2.

int mesNacimiento = 2;

A partir de este momento la variable mesNacimiento almacena el valor 2 y cualquier referencia a ella utiliza este valor. Por ejemplo, si se imprime el valor de la variable por la consola, muestra el valor 2.

printf (mesNacimiento);

C++ permite declarar e inicializar una variable en una sola sentencia.

```
int diaNacimiento = 20;
```

```
int mesNacimiento = 3;
```

Este ejemplo, se declaran las variables diaNacimiento con el valor 20 y mesNacimiento con valor 3.

Si se declara una constante, entonces se debe utilizar el delimitador final y es necesario indicar su valor. En la siguiente declaración se define el valor de la constante pi de tipo double para almacenar un número real.

```
final double PI = 3.1415926536;
```

Es conveniente utilizar nombres apropiados para las variables. El nombre de una variable debe indicar para qué sirve. El nombre de una variable debe explicarse por sí mismo para mejorar la legibilidad del programa.

Si se desea declarar más de una variable a la vez se deben separar las variables en la sentencia de la declaración.

```
int dia = 20, mes = 2, año = 2020;
```

En este ejemplo se declaran tres variables enteras dia, mes y año. La variable día se inicializa con el valor 20, mes con 2 y año con 2020. La siguiente declaración es equivalente a la anterior.

```
int dia = 20;
int  mes = 2;
int año = 2020;
```

```
#include<iostream>
using namespace std;
int suma(int a, int b){
    return a+b;
}
main(){
    int a,b;
    cout<<"ingrese el primer  numero :";cin>>a;
    cout<<"ingrese el segundo  numero :";cin>>b;
    cout<<suma(a,b);
}
```

```
E:\blogger\suma con funciones.exe
ingrese el primer  numero :4
ingrese el segundo  numero :8
12
--------------------------------
```

¡Prepárate que viene lo bueno!

Tipos primitivo: Las variables en C++ pueden ser de un tipo primitivo de datos o una referencia a un objeto. Los tipos primitivos permiten representar valores básicos. Estos tipos se clasifican en números enteros, números reales, caracteres y valores booleanos.

- *Números enteros.* Representan números enteros positivos y negativos con distintos rangos de valores, desde cientos a trillones. Los tipos enteros de C++ son byte, int, short y long.
- *Números reales.* Existen dos tipos de números reales en C++, float y double. La diferencia entre ellos está en el número de decimales que tienen capacidad para expresar y en sus rangos de valores.
- *Caracteres.* El tipo char permite representar cualquier carácter Unicode. Los caracteres Unicode contienen todos los caracteres del alfabeto de la lengua castellana.
- *Booleano.* Se utiliza para representar los valores lógicos verdadero y falso. Solo tiene dos valores true y false.

Tabla de tipos primitivos de C++.

Tipo	Descripción	Valor mínimo y máximo
Byte	Entero con signo	-128 a 127
short	Entero con signo	-32768 a 32767
int	Entero con signo	-2147483648 a 2147483647
long	Entero con signo	-922117036854775808 a +922117036854775807
float	Real de precisión simple	±3.40282347e+38 a ±1.40239846e-45
double	Real de precisión doble	±1.7976931348623157e+309 a ±4.94065645841246544e-324
char	Caracteres Unicode	\u0000 a \uFFFF
boolean	Valores lógicos	true, false

Literales: Se denomina literal a la manera en que se escriben los valores para cada uno de los tipos primitivos.

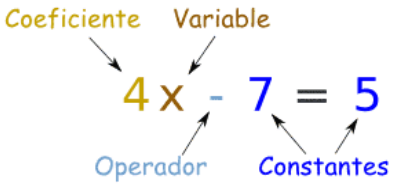
- **Números enteros**. un número entero en c++ se puede escribir en decimal, octal o en hexadecimal. cuando se utiliza el sistema octal es necesario poner el dígito 0 delante del número. si se utiliza el sistema hexadecimal se debe poner 0x delante del número. por ejemplo, el número decimal 10 se puede escribir 012 en octal y 0xa en hexadecimal. los números enteros se supone que pertenecen al tipo int.
- **Números reales**. un número real en c++ siempre debe tener un punto decimal o un exponente. por ejemplo, el número 0.25 se puede expresar también como 2.5e-1. los números reales se supone que pertenecen al tipo double.

- **Booleanos**. los valores lógicos solo pueden ser true y false. se escriben siempre en minúsculas.
- **Caracteres.** los valores de tipo carácter representan un carácter unicode. se escriben siempre entre comillas simples, por ejemplo 'a', 'a', '0', '9'. en c++ un carácter se puede expresar por su código de la tabla unicode en octal o en hexadecimal. los caracteres que tienen una representación especial se indican en la siguiente tabla.

Carácter	Significado
\b	Retroceso
\t	Tabulador
\n	Salto de línea
\r	Cambio de línea
\"	Carácter comilla doble
\'	Carácter comilla simple
\\	Carácter barra hacia atrás

- **Textos**. Un texto en C++ pertenece a la clase String y se expresa como el texto entre comillas dobles. Un texto siempre debe aparecer en una sola línea. Para dividir un texto en varias líneas se debe utilizar el carácter "\n" entre los textos.

  Un texto puede estar vacío o contener uno o más caracteres. Por ejemplo, "Hola Mundo" es un texto de 10 caracteres, mientras que "" es un texto vacío y tiene 0 caracteres. El texto "a" es diferente del carácter 'a' de tipo char.



Operadores: Cada tipo puede utilizar determinados operadores para realizar operaciones o cálculos.

Números enteros. Al realizar una operación entre dos números enteros, el resultado siempre es un número entero. Con los números enteros se pueden realizar operaciones unarias, aditivas, multiplicativas, de incremento y decremento, relacionales, de igualdad y de asignación.

- Una operación unaria permite poner un signo delante: +5, -2.
- Una operación aditiva se refiere a la suma y la resta: 2+3, 5-2.
- Una operación multiplicativa multiplica o divide dos valores: 5*2, 5/2. El operador % calcula el resto de la división entera 5%2.
- Un incremento o decremento aumenta o decrementa en 1 el valor de una variable: ++numero, numero++, --numero, numero--. Si el operador va antes de la variable, primero se realiza la operación y se modifica el valor de la variable. Si el operador va después de la variable, su valor se modifica al final.
- Un operador relacional permite comparar dos valores: >, <, >= y <=. El resultado de la comparación es un valor booleano que indica si la relación es verdadera o falsa.
- Un operador de igualdad compara si dos valores son iguales o no. El operador == devuelve verdadero si los dos valores son iguales, el operador != devuelve verdadero si son diferentes. El resultado de la comparación es un valor booleano que indica si la igualdad o desigualdad es verdadera o falsa.
- Un operador de asignación permite asignar un valor o el resultado de una operación a una variable: =, +=, -=, *=, /=, %=.

Números reales. Con los números reales se aplican los mismos operadores que con los números enteros. Si se realizan operaciones unarias, aditivas o multiplicativas, el resultado es un número real. También se pueden aplicar los operadores relacionales para comparar dos números reales.

Booleanos. Los operadores que se aplican a los valores lógicos son: negación, Y lógico, O lógico.

- LA NEGACIÓN (!) DEVUELVE TRUE SI EL OPERANDO ES FALSE.
- EL Y LÓGICO (&&) DEVUELVE FALSE SI UNO DE LOS OPERANDOS ES FALSE.
- EL O LÓGICO (||) DEVUELVE TRUE SI UNO DE LOS OPERANDOS ES TRUE.

Expresiones: Una expresión permite realizar operaciones entre valores utilizando distintos operadores. Las expresiones son útiles para representar las fórmulas matemáticas que se utilizan para realizar cálculos.

En C++ se pueden definir expresiones tan complejas como sea necesario. Por ejemplo, para convertir una temperatura de grados Fahrenheit a Centígrados se utiliza la fórmula:

C = ((F – 32) * 5) / 9

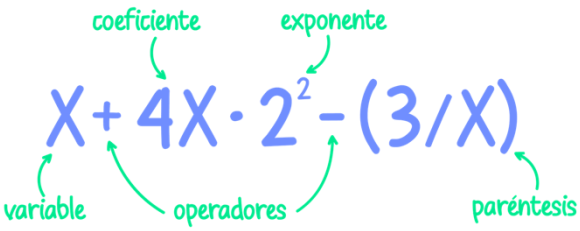
En este ejemplo C representa la temperatura en grados centígrados y F en grados Fahrenheit. La fórmula en Java, utilizando las variables centigrados y fahrenheit de tipo double.

```
centigrados = ((fahrenheit – 32.0) * 5.0)) / 9.0;
```

Toda la expresión se evalúa a un valor. El orden de los cálculos depende del orden de prioridad de los operadores: primero los operadores unarios, después los multiplicativos, de izquierda a derecha, después los operadores aditivos, de izquierda a derecha, después los operadores de relación y por último los operadores de asignación.

Por ejemplo, la expresión x = -3 + 2 * 4 – 12 / 6 + 5 se calcula en el orden siguiente:

Primero se aplica el operador unario (-) a 3 y se obtiene -3. A continuación se evalúan los operadores multiplicativos de izquierda a derecha. Se calcula el producto de 2 * 4 y se obtiene 8, después se divide 12 / 6 y se obtiene 2. Al finalizar estas operaciones la expresión queda x = -3 + 8 – 2 + 5. Por último, se evalúan los operadores aditivos de izquierda a derecha y se obtiene 8.



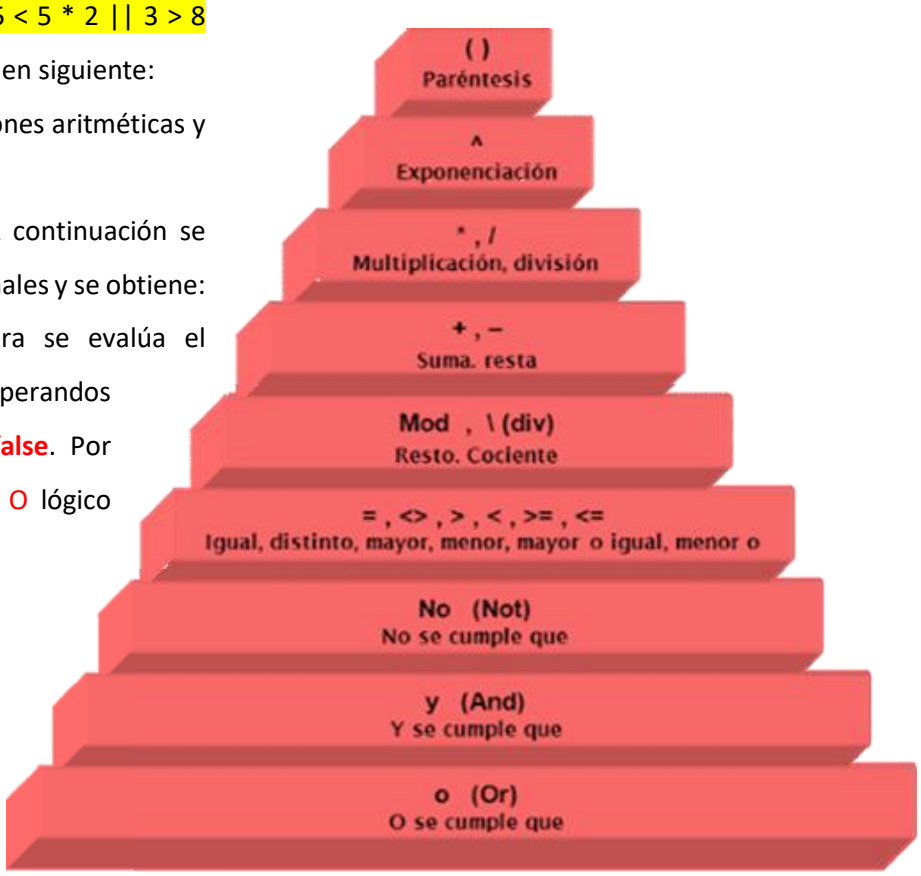
Cuando se desea modificar el orden de prioridad de los operadores es necesario utilizar paréntesis para indicar el orden de evaluación. Por ejemplo, al calcular el valor de y = -3 + 2 * (14 – 2) / 6 + 5 se obtiene 6.

Expresiones aritmético-lógicas: Una expresión aritmético-lógica devuelve un valor lógico verdadero o falso. En este tipo de expresiones se utilizan operadores aritméticos, operadores relacionales y de igualdad. Por ejemplo, una expresión lógica puede ser: (10 – 2) > (5 – 3)

En este ejemplo la expresión aritmético-lógica es verdadera porque el lado derecho de la expresión es mayor que el lado izquierdo.

En una expresión aritmético-lógica se pueden combinar varias expresiones con operadores lógicos. La precedencia de los operadores lógicos es menor que la de los operadores relacionales, por lo que primero se evalúan las desigualdades y después los operadores lógicos. El orden de prioridad de los operadores lógicos es el siguiente: primero la negación, después el Y lógico y por último el O lógico. La prioridad de los operadores de asignación es la menor de todas.

Por ejemplo, la expresión 3 + 5 < 5 * 2 || 3 > 8 && 7 > 6 – 2 se evalúa en el orden siguiente: Primero se evalúan las expresiones aritméticas y se obtiene la expresión lógica: **8 < 10 || 3 > 8 && 7 > 4**. A continuación se evalúan los operadores relacionales y se obtiene: **true || false && true**. Ahora se evalúa el operador Y lógico con los operandos **false && true** y se obtiene **false**. Por último, se evalúa el operador O lógico con los operandos **true || false** y se obtiene **true**, el valor final de la expresión.



Los operadores lógicos && y || se evalúan por cortocircuito. Esto significa que al evaluar a && b, si a es falso, no es necesario evaluar b porque la expresión es falsa. De forma similar, al evaluar a || b, si a es verdadero, no es necesario evaluar b porque la expresión es verdadera.

Jerarquía de operadores

Conversión de tipos: Muchas veces es necesario realizar conversiones de tipos cuando se evalúa una expresión aritmética. Por ejemplo, si después de realizar el cálculo de conversión de grados Fahrenheit a Centígrados se quiere almacenar el resultado en la variable de tipo entero temperatura, es necesario hacer una conversión de tipos. La fórmula en C++, utilizando las variables centigrados y fahrenheit de tipo double.

```
centigrados = ((fahrenheit – 32.0) * 5.0) / 9.0
```

Antes de asignar el valor resultante a la variable temperatura, que almacena un valor entero, es necesario convertir el valor de tipo double de la variable centigrados a int.

```
Int temperatura = (int) centigradros;
```

Palabras reservadas:

En todos los lenguajes de programación existen palabras con un significado especial. Estas palabras son reservadas y no se pueden utilizar como nombres de variables.

La siguiente tabla muestra las palabras reservadas más utilizadas en C++ y otros lenguajes de programación.

abstract	final	public
assert	finally	return
boolean	float	short
break	for	static
byte	if	strictfp
case	implements	super
catch	import	switch
char	instanceof	synchronized
class	int	this
continue	interface	throw
default	long	throws
do	native	transient
double	new	true
else	null	try
enum	package	void
extends	private	volatile
false	protected	while

Actividad sugerida para el estudiante

Modifica el programa para:

1. Cambiar los valores de las variables.
2. Agregar otra operación (resta o división).
3. Mostrar un mensaje diferente si edad es menor de 18.

```cpp
#include <iostream>  // Biblioteca para entrada y salida de datos
using namespace std; // Permite usar cout y cin sin escribir std::

int main() {  // Función principal

    // --- Tipos de datos y variables ---
    int edad = 18;            // Entero
    float altura = 1.75;      // Decimal
    char letra = 'A';         // Un solo carácter
    bool estudiante = true;   // Valor lógico (verdadero o falso)

    // --- Operadores ---
    int suma = edad + 5;      // Operador aritmético (+)
    int multiplicacion = edad * 2;
    bool mayorDeEdad = edad >= 18; // Operador relacional (>=)

    // --- Mostrar resultados ---
    cout << "Edad original: " << edad << endl;
    cout << "Edad + 5: " << suma << endl;
    cout << "Edad multiplicada por 2: " << multiplicacion << endl;
    cout << "¿Es mayor de edad? " << mayorDeEdad << endl;

    return 0; // Fin del programa
}
```

Bibliografía de Lectura 3

Joyanes Aguilar, L. (2010). *Fundamentos de programación en C++.* España: McGraw-Hill

YAÑEZ, L. H. (2014). FUNDAMENTOS DE PROGRAMACION. Madrid, España: Creative Commons. Pag. 01-568

Meza González, J. (2020). Curso de C++. Aprende C++ de una buena vez. Disponible en: https://www.programarya.com/Cursos/C++/Estructura

Programación para principiantes. Disponible en: https://fisiprogramacion.wordpress.com/2014/07/29/c-2-mensajes-por-consola-en-c/

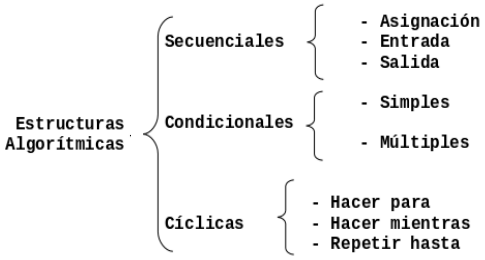
Códigos Qr de acceso a páginas web y archivos pdf.

Lectura No. 4 "Estructuras de Control en C++"

Instrucciones: *Realiza la Lectura 4. "Estructuras de Control en C++" subrayando las ideas principales, y al finalizar realizar las prácticas de cada estructura de control.*

LECTURA 4. "ESTRUCTURAS DE CONTROL EN C++"

Las **estructuras de control** o **construcciones de control** determinan la secuencia en la que se ejecutarán las instrucciones de un programa, se dividen en tres categorías en función del flujo de ejecución:



1. Estructura Secuencial

La estructura secuencial está formada por una secuencia de instrucciones que se ejecutan en orden una a continuación de la otra. Cada una de las instrucciones están separadas por el carácter punto y coma (;). El bloque de sentencias se define por el carácter llave de apertura ({) para marcar el inicio de este, y el carácter llave de cierre (}) para marcar el final.

Ejemplo:

{

instrucción 1;

instrucción 2;

instrucción 3;

.....

instrucción N;

}

```
1   /* Programa: Hola mundo*/
2
3   #include <conio.h>
4   #include <stdio.h>
5
6   int main()
7   {
8       printf( "\n   Hola mundo, Bienvenidos." );
9       printf( "\n\n   Pulse una tecla para salir..." );
10
11      getch(); /* Pausa */
12
13      return 0;
14  }
```

Programa que muestra en la pantalla el mensaje "Hola mundo, además del mensaje "Pulse una tecla para salir..."

Sin embargo, en caso de que el bloque de sentencias este constituido por una única sentencia no es obligatorio el uso de las llaves de apertura y cierre ({ }).

2. Estructura Condicional, Selectiva o Alternativa.

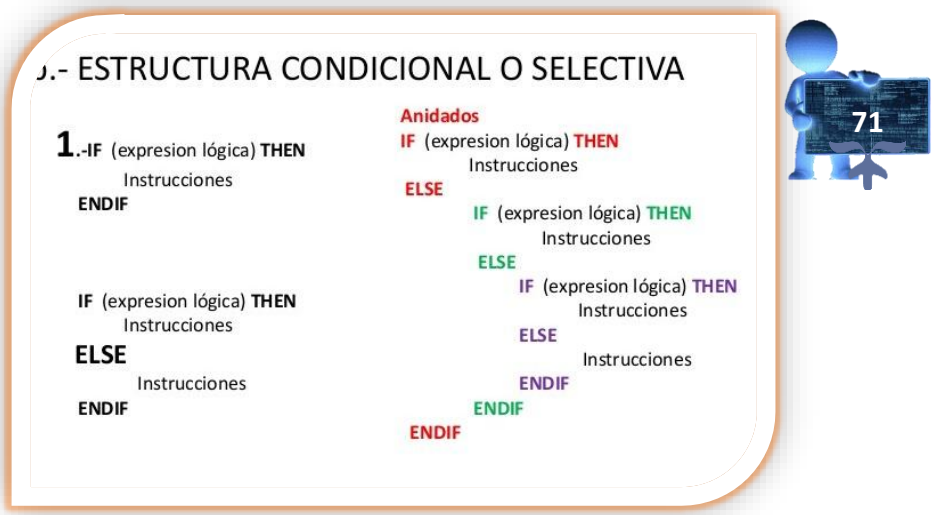
Las estructuras condicionales controlan si una sentencia o bloque de sentencias se ejecutan, en función del cumplimiento o no de una condición o expresión lógica. C++ tiene dos estructuras de control para la selección, **if** y **switch**.

INSTRUCCIÓN IF

Esta instrucción hace que se ejecuten unas sentencias u otras dependiendo del valor que toma una condición. La instrucción if puede ser simple o doble:

Alternativa simple:

```
if (condición)
        instrucción1;

if (condición)
{
        instrucción 1;
        instrucción 2;
        instrucción 3;
}
```



Alternativa doble:

```
if (condición)
        instrucción1;
else
        instrucción2;
if (condición)
{
        instrucción 1;
        instrucción 2;
}
else
{
        instrucción 3;
        instrucción 4;
}
```

```
1   #include <iostream>
2   using namespace std;
3   int main(void)
4   {
5     int num;
6     cout <<"Introduzca numero:";
7     cin >> num;
8     if ((num%2)==0)
9         cout << "PAR" << endl;
10    else
11        cout << "IMPAR" << endl;
12    }
13
```

Programa que lee un número entero por teclado y muestra si es par o impar.

Las instrucciones **if-else** se pueden anidar obteniéndose una **estructura condicional múltiple**:

```
if (condición 1)
 instrucción 1;
  else if (condición 2)
   instrucción 2;
    else if (condición 3)
     instrucción 3;
      else if (condición 4)
       instrucción 4;
       else if (condición 5)
        instrucción 5;
instrucción 6;
…….
```

```
1   #include <iostream>
2   using namespace std;
3   int main(void)
4   {
5       int hora;
6       cout << "\nIntroduzca una hora (entre 0 y 24): ";
7       cin >> hora;
8       if ((hora >= 0) and (hora < 12))
9            cout << "\nBuenos dias\n";
10      else if ((hora >= 12) and (hora < 18))
11                cout << "\nBuenas tardes\n";
12           else if ((hora >= 18) and (hora < 24))
13                     cout << "\nBuenas noches\n";
14                else
15                     cout << "\nHora no válida\n";
16      }
17
```

Programa que lee un número entero que corresponde a una hora y muestra un mensaje según la hora que se haya leído Programa que lee un número entero que corresponde a una hora y muestra un mensaje según la hora que se haya leído

Programa que lee la calificación numérica obtenida por un alumno en un examen y muestra la nota equivalente en texto.

```
#include <iostream>
using namespace std;
int main(void)
{
    unsigned int nota;
    cout << "Introduzca una calificacion numerica entre 0 y 10:";
    cin >> nota;
    cout << "La calificacion del alumno es" << endl;
    if (nota == 10)
    {
       cout << "Matricula de Honor" << endl;
    }
    else if (nota >= 9)
         {
            cout << "Sobresaliente" << endl;
         }
         else if (nota >= 7)
              {
                 cout << "Notable" << endl;
              }
              else if (nota >= 5)
                   {
                      cout << "Aprobado" << endl;
                   }
                   else
                   {
                      cout << "Suspenso" << endl;
                   }
}
```

INSTRUCCIÓN SWITCH

La sentencia switch selecciona una de entre múltiples alternativas. La forma general de esta expresión es la siguiente:

```
switch(variable){
  case 1 : //Acciones 1
    break;
  case 2 : //Acciones 2
    break;
  case 3 : //Acciones 3
    break;
  .
  .
  .
  case N : //Acciones N
    break;
  default: //Acciones default
}
```

```
switch(variable){
    case 'a' : //Acciones a
      break;
    case 'b' : //Acciones b
      break;
    case 'c' : //Acciones c
      break;
    .
    .
    .
    case 'z' : //Acciones z
      break;
    default: //Acciones default
}
```

```
switch (expresión)
{
        case constante1:
            instrucciones;
            break;
        case constante 2:
            instrucciones;
            break;
            . . .
            default:
            instrucciones;
        }
```

```
#include <iostream>
using namespace std;
int main(void)
{
    int A,B, Resultado;
    char operador;
    cout << "Introduzca un numero:";
    cin >> A;
    cout << "Introduzca otro numero:";
    cin >> B;
    cout <<"Introduzca un operador (+,-,*,/):";
    cin >> operador;
    Resultado = 0;
    switch (operador)
    {
            case '-' : Resultado = A - B;
                       break;
            case '+' : Resultado = A + B;
                       break;
            case '*' : Resultado = A * B;
                       break;
            case '/' : Resultado = A / B; //suponemos B!=0
                       break;
            default : cout << "Operador no valido"<< endl;
    }
    cout << "El resultado es: ";
    cout << Resultado << endl;
}
```

Programa que lee dos números y una operación y realiza la operación entre esos números.

En una instrucción switch, expresión debe ser una expresión con un valor entero, y constante1, constante2, ..., deben ser constantes enteras, constantes de tipo carácter o una expresión constante de valor entero. Expresión también puede ser de tipo char, ya que los caracteres individuales tienen valores enteros. Dentro de un case puede aparecer una sola instrucción o un bloque de instrucciones. La instrucción switch evalúa la expresión entre paréntesis y compara su valor con las constantes de cada case. Se ejecutarán las instrucciones de aquel case cuya constante coincida con el valor de la expresión, y continúa hasta el final del bloque o hasta una instrucción que transfiera el control fuera del bloque del switch (una instrucción break, o return). Si no existe una constante igual al valor de la expresión, entonces se ejecutan las sentencias que están a continuación de default si existe (no es obligatorio que exista, y no tiene porqué ponerse siempre al final).

Programa que determina si un carácter leído es o no una vocal. En ese caso como la sentencia a ejecutar por todas las etiquetas case es la misma, esta sentencia se pone una única vez al final:

```
#include <iostream>
using namespace std;
int main(void)
{
    char car;
    cout << "Introduzca un caracter: ";
    cin >> car;
    switch (car)
    {
            case 'a':
            case 'A':
            case 'e':
            case 'E':
            case 'i':
            case 'I':
            case 'o':
            case 'O':
            case 'u':
            case 'U': cout << car << " es una vocal" << endl;
                      break;
            default : cout << car << " no es una vocal" << endl;
    }
}
```

3. Estructuras Repetitivas o Iterativas.

C++ dispone de tres estructuras repetitivas: **while, do-while** y **for**.

INSTRUCCIÓN WHILE.

```
while (condicion)
{
   instrucción 1;
   ..............
   instrucción N;
}
```

```
/*Programa que lee números hasta que se lee un negativo y muestra la
  suma de los números leídos */
#include <iostream>
using namespace std;
int main(void)
{
    int suma, num;
    suma = 0;
    cout << "Introduzca un numero: ";
    cin >> num;
    while (num >= 0)
    {
        suma = suma + num;
        cout << "Introduzca un numero: ";
        cin >> num;
    }
    cout << endl << "La suma es: " << suma << endl;
}
```

Programa que lee números enteros hasta que se lee un número negativo. Se muestra la suma de todos los números leídos excepto el número negativo.

Ejecuta una instrucción o un bloque de instrucciones cero o más veces, dependiendo del valor de la condición. Se evalúa la condición, y si es cierta, se ejecuta la instrucción o bloque de instrucciones y se vuelve a evaluar la condición; pero si la condición es falsa, se pasa a ejecutar la siguiente instrucción después del while.

Instrucción do .. While.

```
do
{
instrucción 1;
.....
instrucción N;
} while (condición);
```

```
1   /* Lee un número entre 1 y 100 */
2   #include <iostream>
3   using namespace std;
4   int main(void)
5   {
6       int numero;
7       do
8       {
9           cout << "Introduzca un numero entre 1 y 100: ";
10          cin >> numero;
11      }
12      while (numero < 1 || numero > 100);
13  }
14
```

Programa que lee un número entero. El número debe estar comprendido entre 1 y 100.

Ejecuta una instrucción o un bloque de instrucciones, una o más veces, dependiendo del valor de la condición. Se ejecuta la instrucción o bloque de instrucciones y a continuación se evalúa la condición. Si la condición es cierta, se vuelve a ejecutar la instrucción o bloque de instrucciones, y si es falsa, pasa a ejecutarse la siguiente instrucción después del do-while.

Instrucción For.

Un bucle for hace que una instrucción o bloque de instrucciones se repitan un número determinado de veces mientras se cumpla la condición.

for (inicialización; condición; incremento/decremento)

{

instrucción 1;

.....

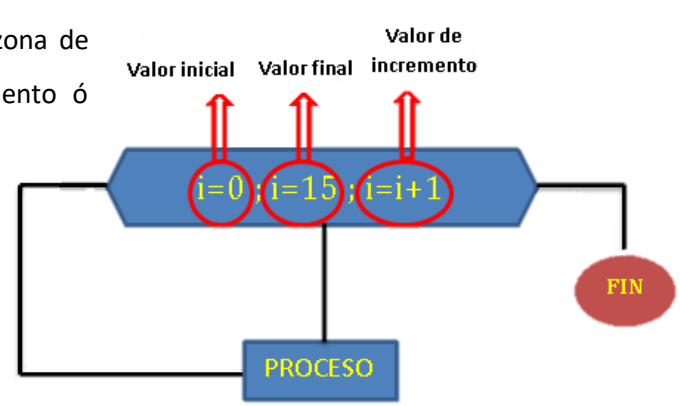
instrucción N;

}

```
1   /* muestra los números de 1 a 10 */
2   #include <iostream>
3   using namespace std;
4   int main(void)
5   {
6       int n;
7       for (n = 1; n <= 10; n++)
8       {
9           cout << n << endl;
10      }
11  }
12
```

Programa que muestra los números del 1 al 10.

A continuación de la palabra for y entre paréntesis debe haber siempre tres zonas separadas por punto y coma: zona de inicialización, zona de condición, zona de incremento ó decremento. En alguna ocasión puede no ser necesario escribir alguna de ellas. En ese caso se pueden dejar en blanco, pero los punto y coma deben aparecer.

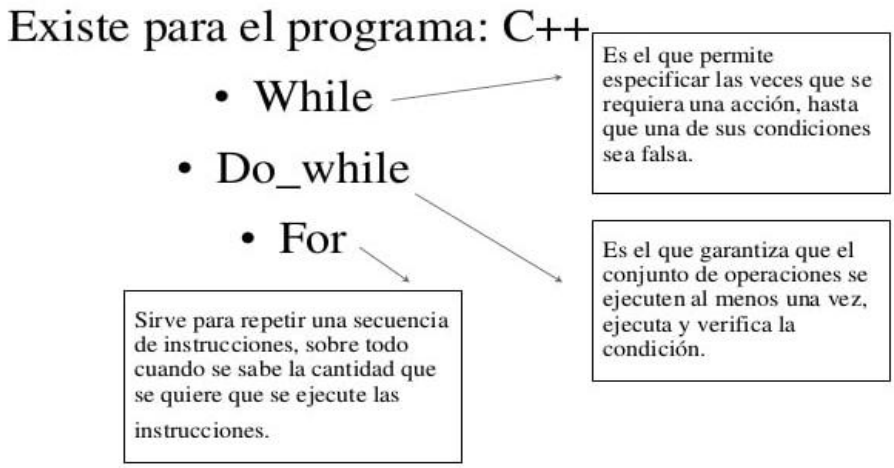


El funcionamiento de un bucle for es el siguiente:

- Se inicializa la variable o variables de control.
- Se evalúa la condición.
- Si la condición es cierta se ejecutan las instrucciones. Si es falsa, finaliza la ejecución del bucle y continúa el programa en la siguiente instrucción después del for.
- Se actualiza la variable o variables de control (incremento/decremento).
- Se pasa al punto 2).

Esta instrucción es especialmente indicada para bucles donde se conozca el número de repeticiones que se van a hacer.

Como regla práctica podríamos decir que las instrucciones while y do-while se utilizan generalmente cuando no se conoce a priori el número de pasadas, y la instrucción for se utiliza generalmente cuando sí se conoce el número de pasadas.



Ejercicios sugeridos para los estudiantes

Estructura if (condicional)

- Solicita la edad de una persona e indica si es mayor o menor de edad.
- Pide un número y muestra si es positivo, negativo o cero.
- Ingresa la calificación de un alumno e indica si aprobó (≥ 6) o reprobó.
- Solicita dos números e indica cuál es mayor.

Estructura switch

- Pide un número del 1 al 7 y muestra el día de la semana correspondiente.
- Solicita un número del 1 al 3 y muestra el nombre de un color (1=Rojo, 2=Verde, 3=Azul).
- Ingresa una opción (1, 2 o 3) para elegir una operación básica (+, -, *) y muestra el resultado con dos números dados.

Ciclo for

- Muestra los números del 1 al 10.
- Imprime la tabla de multiplicar de un número dado.
- Calcula la suma de los números del 1 al 100.
- Muestra todos los números pares del 1 al 20.

Ciclo while

- Pide números al usuario hasta que ingrese un número negativo.
- Solicita una contraseña y repite hasta que sea correcta.
- Muestra los números del 1 al 10 usando un ciclo while.
- Cuenta cuántos números se ingresaron antes de escribir cero.

Ciclo do-while

- Solicita un número y repite hasta que sea mayor que 0.
- Muestra un menú de opciones hasta que el usuario elija salir.
- Pide una calificación hasta que esté en el rango de 0 a 10.
- Calcula la suma de números ingresados hasta que el usuario decida terminar.

Bibliografía de Lectura 4

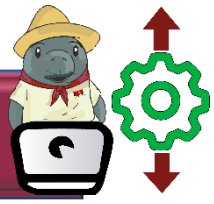
JOYANES AGUILAR, L. (2010). FUNDAMENTOS DE PROGRAMACION EN C++. ESPAÑA: McGRAW-HILL. Pág. 01-802

YAÑEZ, L. H. (2014). FUNDAMENTOS DE PROGRAMACION. Madrid, España: Creative Commons. Pag. 01-568

García Hernández, G. (s/f). Programación C++. Disponible en: http://ejercicioscpp.blogspot.com/2012/11/estructuras-de-control-en-c.html

Olimpiada de Informática del estado de Jalisco. Disponible en: http://www.omijal.org/pagina_c/control.html

Códigos Qr de acceso a páginas web y archivos pdf.



Estructura SIMPLE-SECUENCIAL
Práctica No. 01 "Promedio de 3 Calificaciones"

PROPÓSITO: Utiliza la estructura simple y analiza sus ventajas y desventajas, teniendo en cuenta su utilidad en su vida escolar y profesional.

CONOCIMIENTOS

- ✓ Estructura Simple
- ✓ Uso del editor Dev C++
- ✓ Librerías para utilizar
- ✓ Función principal
- ✓ Compilar y ejecutar
- ✓ Guardar código

DESARROLLO

1. Realiza un programa de estructura secuencial, que solicite escribir tres calificaciones y a partir de ello realice el cálculo del promedio de ellas.
2. Abre el editor del Lenguaje C++, Dev-**C**++.
3. Escribe el código en el editor con apoyo de tu profesor.

4. Al terminar el código guarda el archivo en tu carpeta con la siguiente estructura: Practica01, siglas de tu nombre, grupo y turno. Ejemplo: **"Practica01_LYLRHV"** donde LYLR son las siglas de tu nombre, H es el grupo y V el turno para Matutino y V vespertino.
5. Posteriormente compilamos el programa, presiona el botón de compilar  o utilizar las teclas Ctrl-F9.
6. A continuación, ejecutamos el programa, presiona el botón de ejecutar  y se genera el archivo ejecutable de nuestro programa.

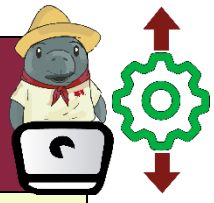
```
Ingresa la primer calificacion:
10


Ingresa la segunda calificacion:
3


Ingresa la tercer calificacion:
7



El promedio de las tres calificaciones es:6.66667
Presione una tecla para continuar . . .
```

7. **CONCLUSIONES:** Al finalizar la práctica, con tus palabras contesta las siguientes preguntas:

   ¿Qué aprendí?

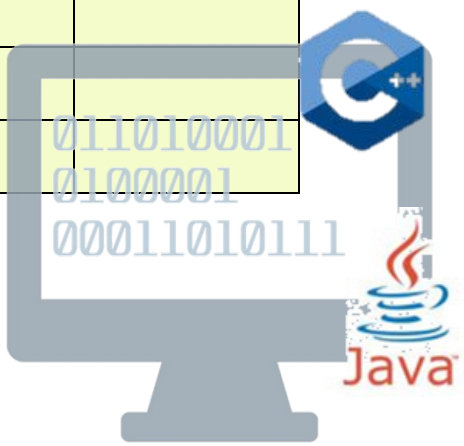
   ¿Dónde lo aplico en mi vida cotidiana y académica?

INSTRUMENTO DE EVALUACIÓN
LISTA DE COTEJO
Práctica 01. "Promedio de 3 Calificaciones"

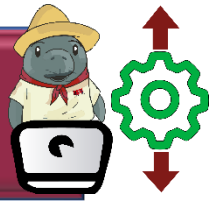


DATOS GENERALES	
Nombre(s) del alumno(s)	**Matricula(s)**
Producto: Programa de Promedio de 3 Calificaciones.	**Fecha**
Submódulo I: Programación en C++	**Periodo: 2026 – 2027A**
Nombre del docente	**Firma del docente**

CRITERIO	INDICADORES	VALOR OBTENIDO		PONDERACIÓN	CALIF	OBSERVACIONES Y/O SUGERENCIAS DE MEJORA
		SÍ	NO			
1	Codifica correctamente la secuencia lógica del desarrollo del programa.			**2**		
2	Identifica y aplica correctamente las librerías e instrucciones.			**2**		
3	Guarda el archivo con la nomenclatura solicitada.			**1**		
4	Compila y ejecuta el programa, mostrando en pantalla el resultado del cálculo del promedio de tres calificaciones.			**3**		
5	Entrega en tiempo y forma.			**1**		
6	Presenta conclusiones de la práctica.			**1**		
				CALIFICACIÓN		



Estructura CONDICIONAL O DECISIÓN
Práctica No. 02 "Determinar de Tres Números Dados Cuál es el Mayor y Cuál el Menor"



PROPÓSITO: Utiliza la estructura condicional y analiza sus ventajas y desventajas, teniendo en cuenta su utilidad en su vida escolar y profesional.

CONOCIMIENTOS

- ✓ Estructura IF ELSE
- ✓ Variable inicial
- ✓ Variable final
- ✓ Condición

DESARROLLO

1. Realiza utilizando la estructura condicional, un programa que solicite el ingreso de tres números cualquiera, y determine cuál de ellos es el mayor y cuál el menor.
2. Abre el editor del Lenguaje C++, Dev-**C**++.
3. Escribe el código en el editor con apoyo de tu profesor.
   Al terminar el código guarda el archivo en tu carpeta con la siguiente estructura: Practica02, siglas de tu nombre, grupo y turno. Ejemplo: **"Practica02_CAMCGV"** donde CAMC son las siglas de tu nombre, G es el grupo y M el turno para Matutino y V vespertino.
4. Posteriormente compilamos el programa, presiona el botón de compilar  o utilizar las teclas Ctrl-F9.

5. A continuación, ejecutamos el programa, presiona el botón de ejecutar y se genera el archivo ejecutable de nuestro programa.

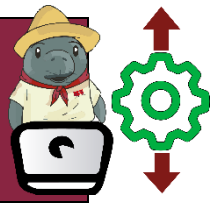
```
Ingresar tres numeros distintos:
Ingrese un numero: 6
Ingrese un numero: 3
Ingrese un numero: 10
El mayor es: 10
El menor es: 3
--------------------------------
Process exited with return value 0
Press any key to continue . . .
```

7. **CONCLUSIONES:** Al finalizar la práctica, con tus palabras contesta las siguientes preguntas:

¿Qué aprendí?

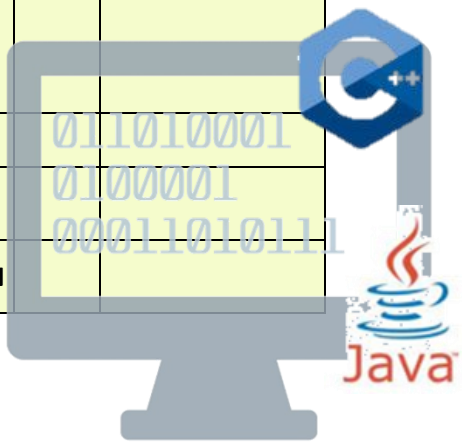
¿Dónde lo aplico en mi vida cotidiana y académica?

INSTRUMENTO DE EVALUACIÓN
LISTA DE COTEJO
Práctica 02. "Determinar de Tres Números Dados Cuál es el Mayor Cuál el Menor"

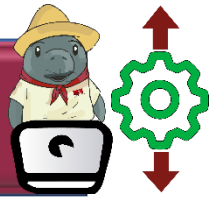


DATOS GENERALES	
Nombre(s) del alumno(s)	**Matricula(s)**
Producto: Programa de 3 Números Dados Cuál es el Mayor y Cuál el Menor.	**Fecha**
Submódulo I: Programación en C++	**Periodo: 2026 – 2027A**
Nombre del docente	**Firma del docente**

CRITERIO	INDICADORES	VALOR OBTENIDO		PONDERACIÓN	CALIF	OBSERVACIONES Y/O SUGERENCIAS DE MEJORA
		SÍ	NO			
1	Codifica correctamente la secuencia lógica del desarrollo del programa.			**2**		
2	Identifica y aplica correctamente las librerías e instrucciones.			**2**		
3	Guarda el archivo con la nomenclatura solicitada.			**1**		
4	Compila y ejecuta el programa, mostrando en pantalla el resultado de determinar de 3 números cual es el mayor y cual el menor.			**3**		
5	Entrega en tiempo y forma.			**1**		
6	Presenta conclusiones de la práctica.			**1**		
	CALIFICACIÓN					



Estructura repetitiva WHILE
Práctica No. 03 "Generar los Números Pares Entre 0 y 100"



PROPÓSITO: Identifica la estructura While y aplica las ventajas de utilizar la sentencia While dentro de los programas con sentencias repetitivas, para el desarrollo de programas en su vida escolar y profesional.

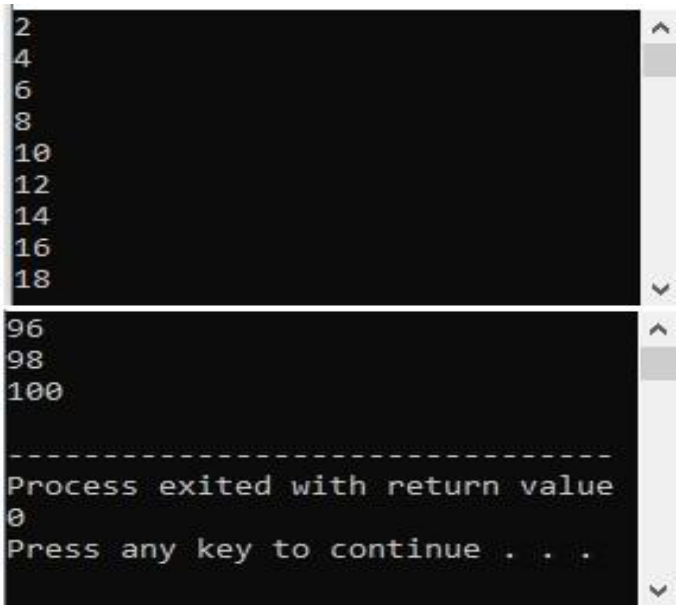
CONOCIMIENTOS

- ✓ Estructura While
- ✓ Variable inicial
- ✓ Variable final
- ✓ Condiciones(booleanas)
- ✓ Contador (incremento)

DESARROLLO

1. Realiza utilizando la estructura repetitiva, un programa que use los números del 0 al 100 para elegir los números pares entres estos y los imprima en pantalla.
2. Abre el editor del Lenguaje C++, Dev-**C**++.
3. Escribe el código en el editor con apoyo de tu profesor.

4. Al terminar el código guarda el archivo en tu carpeta con la siguiente estructura: Practica03, siglas de tu nombre, grupo y turno. Ejemplo: **"Practica03_JAFPGM"** donde JAFP son las siglas de tu nombre, G es el grupo y M el turno para Matutino y V vespertino.
5. Posteriormente compilamos el programa, presiona el botón de compilar  utilizar las teclas Ctrl-F9.
6. A continuación, ejecutamos el programa, presiona el botón de ejecutar  y se genera el archivo ejecutable de nuestro programa.

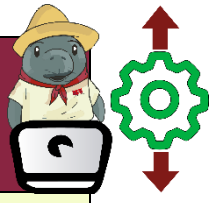


7. **CONCLUSIONES:** Al finalizar la práctica, con tus palabras contesta las siguientes preguntas:

   ¿Qué aprendí?

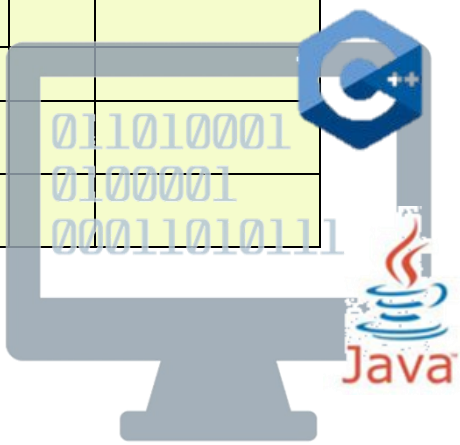
   ¿Dónde lo aplico en mi vida cotidiana y académica?

INSTRUMENTO DE EVALUACIÓN
LISTA DE COTEJO
Práctica 03. "Generar los Números Pares Entre 0 y 100"

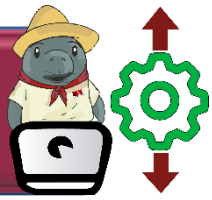


DATOS GENERALES	
Nombre(s) del alumno(s)	**Matricula(s)**
Producto: Programa Generar los Números Pares Entre 0 y 100.	**Fecha**
Submódulo I: Programación en C++	**Periodo: 2026 – 2027A**
Nombre del docente	**Firma del docente**

CRITERIO	INDICADORES	VALOR OBTENIDO		PONDERACIÓN	CALIF	OBSERVACIONES Y/O SUGERENCIAS DE MEJORA
		SÍ	NO			
1	Codifica correctamente la secuencia lógica del desarrollo del programa.			**2**		
2	Identifica y aplica correctamente las librerías e instrucciones.			**2**		
3	Guarda el archivo con la nomenclatura solicitada.			**1**		
4	Compila y ejecuta el programa, mostrando en pantalla el resultado de generación de números pares entre 0 y 100.			**3**		
5	Entrega en tiempo y forma.			**1**		
6	Presenta conclusiones de la práctica.			**1**		
	CALIFICACIÓN					



Estructura repetitiva FOR
Práctica No. 04 "Calcular el Promedio de 3 Calificaciones para N Estudiantes"



PROPÓSITO: Utiliza la estructura For para los programas con sentencias repetitivas y analiza sus ventajas y desventajas, teniendo en cuenta su utilidad en su vida escolar y profesional.

CONOCIMIENTOS

- ✓ Estructura For
- ✓ Variable inicial
- ✓ Variable final
- ✓ Condiciones(booleanas)
- ✓ Contador (incremento)

DESARROLLO

1. Realiza utilizando la estructura repetitiva, un programa que solicite el ingreso de tres números cualquiera, y determine cuál de ellos es el mayor y cual el menor.
2. Abre el editor del Lenguaje C++, Dev-**C**++.
3. Escribe el código en el editor con apoyo de tu profesor.

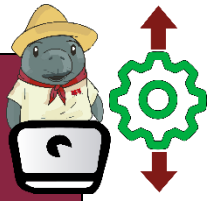
4. Al terminar el código guarda el archivo en tu carpeta con la siguiente estructura: Practica04, siglas de tu nombre, grupo y turno. Ejemplo: **"Practica04_JARHHV"** donde JARH son las siglas de tu nombre, H es el grupo y V el turno para Matutino y V vespertino.
5. Posteriormente compilamos el programa, presiona el botón de compilar  o utilizar las teclas Ctrl-F9.
6. A continuación, ejecutamos el programa, presiona el botón de ejecutar  y se genera el archivo ejecutable de nuestro programa.


```
++++ Calcular el promedio de 3 calificaciones para N estudiantes ++++

CUANTOS ESTUDIANTES ?  2
ALUMNO No.  1
Nota 1: 5

Nota 2: 6

Nota 3: 7

PROMEDIO 6

ALUMNO No.  2
Nota 1: 10

Nota 2: 9

Nota 3: 9

PROMEDIO 9.33333


--------------------------------
Process exited with return value 0
Press any key to continue . . .
```

7. **CONCLUSIONES:** Al finalizar la práctica, con tus palabras contesta las siguientes preguntas:

   ¿Qué aprendí?

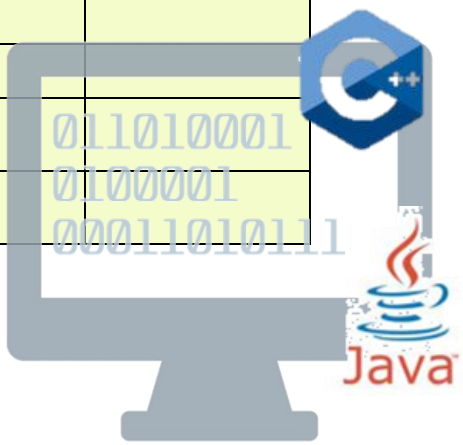
   ¿Dónde lo aplico en mi vida cotidiana y académica?



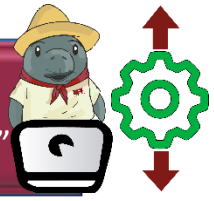
INSTRUMENTO DE EVALUACIÓN
LISTA DE COTEJO
Práctica 04. "Calcular el Promedio de 3 Calificaciones para N Estudiantes"

DATOS GENERALES	
Nombre(s) del alumno(s)	**Matricula(s)**
Producto: Programa Calcular el Promedio de 3 Calificaciones para N Estudiantes.	**Fecha**
Submódulo I: Programación en C++	**Periodo: 2026 – 2027A**
Nombre del docente	**Firma del docente**

CRITERIO	INDICADORES	VALOR OBTENIDO		PONDERACIÓN	CALIF	OBSERVACIONES Y/O SUGERENCIAS DE MEJORA
		SI	NO			
1	Codifica correctamente la secuencia lógica del desarrollo del programa.			**2**		
2	Identifica y aplica correctamente las librerías e instrucciones.			**2**		
3	Guarda el archivo con la nomenclatura solicitada.			**1**		
4	Compila y ejecuta el programa, mostrando en pantalla el resultado del Cálculo del Promedio de 3 Calificaciones Para N Estudiantes.			**3**		
5	Entrega en tiempo y forma.			**1**		
6	Presenta conclusiones de la práctica.			**1**		
				CALIFICACIÓN		



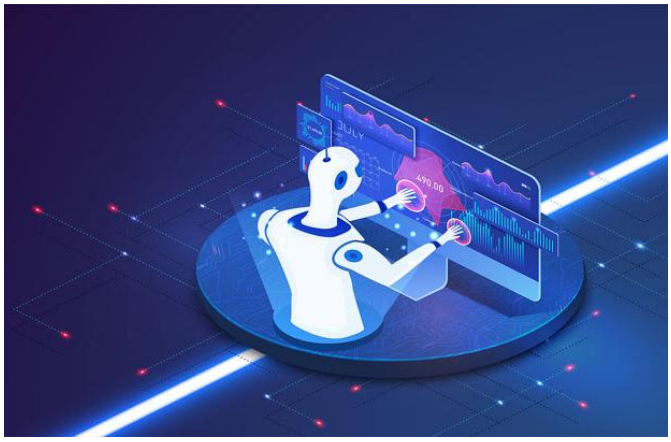
Estructura repetitiva DO-FOR (Ciclos Anidados)
Práctica No. 05 "Dado un Numero Entero N, Calcular su Factorial"



PROPÓSITO: Utiliza los ciclos anidados con las estructuras repetitivas do y FOR, para la resolución de problemas en los programas, teniendo en cuenta su utilidad tanto en su vida escolar y profesional.

CONOCIMIENTOS

- ✓ Estructura repetitiva For
- ✓ Estructura repetitiva Do.While
- ✓ Ciclos Anidados
- ✓ Variable inicial
- ✓ Variable final
- ✓ Condiciones(booleanas)
- ✓ Contador (incremento)



DESARROLLO

1. Realiza el siguiente programa en el lenguaje de C++, dado un número entero para n, calcular su factorial (n!). Utiliza las estructuras repetitivas do-while y for.
2. Abre el editor del Lenguaje C++, Dev-**C**++.
3. Escribe el código en el editor con apoyo de tu profesor.

```
1  //programa para calcular el factorial de un número
2  #include <iostream>
3  using namespace std;
4  int main()
5  {
6     int n,i;
7     char opc;
8     long double factorial; // se declara long double para poder representar números grandes
9     factorial=1;
10
11     cout << "Calcular el Factorial para N numero "<<endl;
12
13    do{          // inicio del ciclo do
14
15    cout <<endl<< "Introduce un numero: ";
16    cin >> n;
17
18     if(n<0)     // condición SI es un numero negativo
19        cout << "Para los numeros negativos, Factorial indefinido "<<endl;
20
21     else if (n==0) //condicion SI es un numero=0
22        cout<<"Factorial de O! = 1"<<endl;
23
24     else         // los numeros del 1 en adelante
25     {
26           cout<<n<<"! = ";
27        for(i=n;i>=1;i--)
28           cout<<i<<" * ";
29        for(i=1;i<=n;i++)
30            factorial = factorial * i;
31            cout << endl << "Factorial de " << n << " -> " << factorial << endl;
32     }
33    cout <<endl<< "DESEA CONTINUAR [S/N] ";
34    cin >> opc;
35     }while(opc=='S' || opc=='s');  // hacer-mientras Elija S para continuar
36     return 0;
37 }
```

4. Al terminar el código guarda el archivo en tu carpeta con la siguiente estructura: Practica05, siglas de tu nombre, grupo y turno. Ejemplo: **"Practica05_JGRSGM"** donde JGRS son las siglas de tu nombre, G es el grupo y M el turno para Matutino y V vespertino.
5. Posteriormente compilamos el programa, presiona el botón de compilar  o utilizar las teclas Ctrl-F9.
6. A continuación, ejecutamos el programa, presiona el botón de ejecutar  y se genera el archivo ejecutable de nuestro programa.

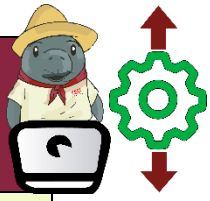
```
Calcular el Factorial para N numero

Introduce un numero: 5
5! = 5 * 4 * 3 * 2 * 1 *
Factorial de 5 -> 120

DESEA CONTINUAR [S/N] s

Introduce un numero: -3
Para los numeros negativos, Factorial indefinido

DESEA CONTINUAR [S/N] s

Introduce un numero: 0
 Factorial de O! = 1

DESEA CONTINUAR [S/N] s

Introduce un numero: 3
3! = 3 * 2 * 1 *
Factorial de 3 -> 720

DESEA CONTINUAR [S/N] n

--------------------------------
Process exited with return value 0
Press any key to continue . . .
```

7. **CONCLUSIONES:** Al finalizar la práctica, con tus palabras contesta las siguientes preguntas:

   ¿Qué aprendí?

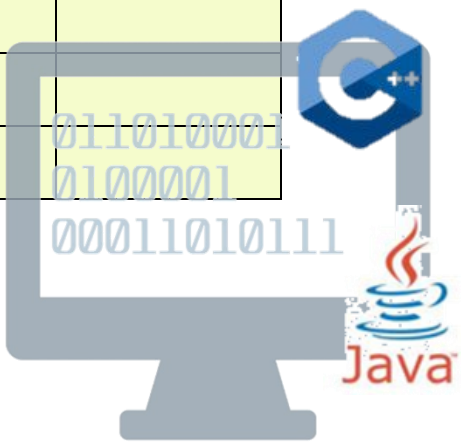
   ¿Dónde lo aplico en mi vida cotidiana y académica?

INSTRUMENTO DE EVALUACIÓN
LISTA DE COTEJO
Práctica 05. "Dado un Número Entero N, Calcular Su Factorial (n!) "



DATOS GENERALES	
Nombre(s) del alumno(s)	**Matricula(s)**
Producto: Programa Dado un Número Entero, Calcular Su Factorial.	**Fecha**
Submódulo I: Programación en C++	**Periodo: 2026 – 2027A**
Nombre del docente	**Firma del docente**

CRITERIO	INDICADORES	VALOR OBTENIDO		PONDERACIÓN	CALIF	OBSERVACIONES Y/O SUGERENCIAS DE MEJORA
		SÍ	NO			
1	Codifica correctamente la secuencia lógica del desarrollo del programa.			**2**		
2	Identifica y aplica correctamente las librerías e instrucciones.			**2**		
3	Guarda el archivo con la nomenclatura solicitada.			**1**		
4	Compila y ejecuta el programa, mostrando en pantalla el resultado de un numero entero N y Calcula el Factorial.			**3**		
5	Entrega en tiempo y forma.			**1**		
6	Presenta conclusiones de la práctica.			**1**		
				CALIFICACIÓN		



Lectura No. 5 "Procedimientos y Funciones"

Instrucciones: *Realiza la Lectura 5. "Procedimientos y Funciones" subrayando las ideas principales, y al finalizar realiza la Actividad 3 – Procedimientos y Funciones: enumera el código.*

LECTURA 5. "PROCEDIMIENTOS Y FUNCIONES"

Los procedimientos y funciones son conjuntos de instrucciones agrupadas, que nos permiten separar un programa complejo en pequeños subprogramas con una determinada tarea. Esto nos ayudará a estructurar un programa de forma que sea más fácil de entender y editar. Tienen una estructura muy parecida; la única diferencia entre ellas es que los **procedimientos** son de tipo **«void»** (que significa vació, por lo tanto, no devuelve nada), en cambio, las **funciones** tienen un tipo determinado: el tipo de valor a devolver al subprograma que la ha llamado, por ejemplo «bool», «int», «char», etc., además han de contener la instrucción «return» junto a la variable o dato que queramos devolver. (Este dato ha de ser del mismo tipo con el que has definido la función).

Una función hace ciertos cálculos y devuelve un valor, y un procedimiento es una secuencia de instrucciones o comandos y no devuelve ningún valor.

PROCEDIMIENTOS

No hacen uso de la sentencia return para devolver valores y no tienen tipo específico, solo *void*.

SINTAXIS

```
void ([tipo nombreArgumento,[tipo nombreArgumento]...])
{
   /*
      * Bloque de instrucciones
   */
}
```

Ejemplo.

Programa que pide dos valores e intercambia entre ellas su valor

PROCEDIMIENTOS_INTERC.cpp

```
1   #include<iostream>
2
3   using namespace std;
4
5   void intercambia(int&a,int&b){//procedimiento
6
7   int aux;
8   aux=a;
9   a=b;
10  b=aux;
11  }
12
13  int main(){
14
15  int a,b;
16  cout<<"Introduce dos valores : "<<endl;
17  cout<<"A : ";
18  cin >> a;
19  cout<<"B : ";
20  cin >> b;
21
22  intercambia(a,b) ; //llamar al procedimiento intercambia
23
24  cout<<"Intercambia los valores : "<<endl;
25  cout<<"A : "<< a <<endl<<"B : "<< b<<endl;
26  }
```

El signo «*&*» se utiliza para pasar por referencia las variables. Esto sirve para poder modificar las variables y que guarde la modificación, para que, al volver al programa principal o a otro subprograma, estas variables mantengan los cambios realizados. En el caso de no poner este signo las variables las pasas por valor, es decir, que puedes usarlas y modificarlas, pero no guardarán los cambios realizados en ellas).

```
Introduce dos valores :
A : 10
B : 1000
Intercambia los valores :
A : 1000
B : 10

--------------------------------
Process exited with return value 0
Press any key to continue . . .
```

FUNCIONES

Hacen uso de la sentencia Return para devolver un valor.

SINTAXIS

tipo nombreFuncion([tipo nombreArgumento,[tipo nombreArgumento]...])

{

/*

* Bloque de instrucciones

*/

return valor;

}

Ejemplo.

Programa que Pide dos números y determina cual es el mayor.

```
1  //Pedir dos numero y determinar el número mayor
2  #include<iostream>
3  using namespace std;
4
5  int mayor(int a,int b){//función para determinar el num mayor
6
7   if(a > b) return a;
8   else return b;
9
10 }
11
12 int main(){
13
14 int a,b, r;
15
16 cout<<"Introduce dos numeros : "<<endl;
17 cout<<"Num1 : ";
18 cin >> a;
19 cout<<"Num2 : ";
20 cin >> b;
21
22 r = mayor(a,b) ; //llama a la función
23
24 cout<<"El numero mayor es : "<< r <<endl;
25
26 }
```

```
Introduce dos numeros :
Num1 : 300
Num2 : 5
El numero mayor es : 300

--------------------------------
Process exited with return value 0
Press any key to continue . . .
```

En este ejemplo, el programa principal crea la-s variables *a* y *b* para introducir los valores y la variable *r* para guardar el resultado de la función.

La función sólo hace una comparación entre *a* y *b* para saber cuál es mayor de los dos.

Bibliografía de Lectura 5 Procedimientos y Funciones

Joyanes Aguilar, L. (2010). Fundamentos De Programación En C++. España: Mcgraw-Hill. Pág.201- 225

Hernández, Uriel (14 de marzo del 2012). *Tutorial C++ 17*. Introducción a Funciones. https://www.youtube.com/watch?v=ZYCTqYvDEI8

Programar-Fácil. (7 de diciembre del 2011). *Tutorial de C++ en Español # 1*. Funciones. https://www.youtube.com/watch?v=Lm5Q6cEsvtc

Meza Contreras, Juan David (2012-2020). *Programar Ya* - Curso de C++. Obtenido de https://www.programarya.com/Cursos/C++/Funciones

Torres Martí, Sergi (16 de abril del 2014). *STM-Blog*. Procedimientos y Funciones en C++. http://stmblog.com/programacion/procedimientos-y-funciones/

Códigos Qr de acceso a pagina web y archivos pdf.

Actividad 03.- Procedimientos y funciones: Enumera y escribe el código

Instrucciones:

1. Lee y analiza el código en lenguaje de C++ dentro de los rectángulos, y con el apoyo de tu maestro/a, enumera los círculos de cada rectángulo en el orden lógico que tendría el programa.

2. Luego escribe el código en una secuencia lógica en tu guía. Programa con funciones

.

3. A continuación, escribe y comprueba su funcionamiento dentro del editor del lenguaje C++ (compilar y ejecutar).

4. Al terminar el código guarda el archivo en tu carpeta con la siguiente estructura: Actividad05, siglas de tu nombre, grupo y turno. Ejemplo: **"Actividad03_MEPAFM"** donde MEPA son las siglas de tu nombre, F es el grupo y M el turno para Matutino y V vespertino.

Programa: Calculadora con funciones básicas de sumar, restar, multiplicar, dividir.

```
Introduce dos Numeros :
100
3000
ELIGE UNA OPCION
1 .- SUMA
2 .- RESTA
3 .- MULTIPLICACION
4 .- DIVISION
0 .- SALIR
3

 MULTIPLICACION = 300000
 MUCHAS GRACIAS POR USAR MI CALCULADORA
--------------------------------
Process exited with return value 0
Press any key to continue . . .
```

ENUMERA EL ORDEN LÓGICO DEL PROGRAMA

- Encabezado
- Declaración de funciones
- Programa principal
- Llamado a las funciones

Programa: Calculadora con funciones básicas de sumar, restar, multiplicar.

```
if(opcion==1){
cout<<"\n SUMA = "<<suma(n1,n2); //utilizamos la funcion suma
}
```

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

```
int main(){
double n1,n2; //variables para operar
int opcion; //variable para la opcion elegida

cout<<"Introduce dos Numeros :"<<endl;
cin>>n1>>n2;
```

```
do{
cout<<"ELIGE UNA OPCIÓN"<<endl;
cout<<"1 .- SUMA"<<endl;
cout<<"2 .- RESTA"<<endl;
cout<<"3 .- MULTIPLICACION"<<endl;
cout<<"4 .- DIVISION"<<endl;
cout<<"0 .- SALIR"<<endl;
cin>>opcion;
}while(opcion>4);
```

```
double resta(double a,double b){
        return a-b;
}
```

```
if(opcion==2){
    cout<<"\n RESTA = "<<resta(n1,n2); //utilizamos la funcion resta
}
```


```
cout<<"\n MUCHAS GRACIAS POR USAR MI CALCULADORA";

}
```

```
double multiplica(double a,double b){
        return a*b;
}
```

```
if(opcion==3){
    cout<<"\n MULTIPLICACION = "<<multiplica(n1,n2);
}
```

```
double suma(double a,double b){
        return a+b;
}
```

```
double divide(double a,double b){
    if(b!=0){
            return a/b;
    }
    else   cout<<"ERROR: El divisor es cero";
}
```

```
if(opcion==4){
    cout<<"\n DIVISION = "<<divide(n1,n2); //utilizamos la funcion divide
}
```

ESCRIBE EL PROGRAMA: Calculadora con funciones básicas para sumar, restar, multiplicar y dividir.

PROGRAMA RESUELTO

Calculadora con funciones básicas de sumar, restar, multiplicar.

INSTRUMENTO DE EVALUACIÓN
LISTA DE COTEJO
Actividad 03. Procedimientos y funciones: Enumera y escribe el código

DATOS GENERALES	
Nombre(s) del alumno(s)	**Matricula(s)**
Producto: Programa	**Fecha**
Submódulo I: Programación en C++	**Periodo: 2026 – 2027A**
Nombre del docente	**Firma del docente**

CRITERIO	INDICADORES	VALOR OBTENIDO		PONDERACIÓN	CALIF	OBSERVACIONES Y/O SUGERENCIAS DE MEJORA
		SÍ	NO			
1	Entrego en tiempo y Forma.			**2**		
2	Agrego la información correcta.			**2**		
3	Se aprecia el conocimiento obtenido del tema.			**1**		
4	Presenta el código perfectamente ordenado.			**2**		
5	Utiliza la tecnología para codificar y ejecutar el programa.			**2**		
6	Mantienen interés en la comprensión de la actividad.			**1**		
				CALIFICACIÓN		

PROGRAMANDO TU SALUD

Propósito:

Elaborar un programa creativo en C++, en formato físico o digital, usando el Dev C++ o Editor en C++ sugerido por el docente, proponiendo procedimientos y funciones, que permitan a través de la cantidad de glucosa de una persona (detección de diabetes), determinar los niveles de glucosa (Normal, Prediabético y diabético) y aportar recomendaciones nutricionales por cada nivel de glucosa, en equipos de 5 integrantes para socializarlo ante el grupo.

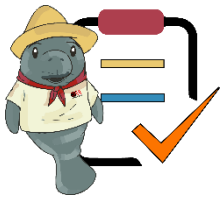
Instrucciones: *Con los conocimientos que has adquirido a lo largo del submódulo, es momento de diseñar en equipo, un programa en el que se hace conciencia sobre la salud, a través de los hábitos alimenticios y los peligros de la diabetes.*

¿Qué debemos hacer?

Considera la sintaxis y las estructuras de control estudiadas en las lecturas 3. Sintaxis e instrucciones básicas en C++, 4. Estructuras de control en C++. Para después utilizar el compilador de tu preferencia como software para desarrollar su propuesta de programa, no olviden realizar su investigación acerca de la diabetes, que incluya los niveles y las recomendaciones nutricionales para cada nivel.

Al final, realicen y socialicen la reflexión a través de las preguntas del conflicto cognitivo.

1. Define que es un programa en C++.
2. ¿Cuáles son las condiciones para determinar los niveles de glucosa y poder dar recomendaciones?
3. ¿Qué es la programación estructurada?
4. ¿Qué es una estructura de control y cuales aplicaste para la resolución de la situación didáctica?
5. ¿Cómo identificas una condición anidada en la resolución del problema?
6. ¿Qué recomendaciones darías para adquirir un hábito de alimentación balanceada?



Situación Didáctica: "Programando Tu Salud"

Criterios de evaluación	RÚBRICA					
	Descriptores					
Criterio	Excelente 100 - 95	Muy bueno 94 - 80	Aceptable 79 - 60	Suficiente 59 - 50	Insuficiente 40 o menos	PTOS 100
FUNCIONAMIENTO (50% indispensable para evaluar el resto del programa)	Equivalencia 20 puntos	Equivalencia 15 puntos	Equivalencia 10 puntos	Equivalencia 5 puntos	Equivalencia 0 puntos	20
	Al Compilar y al ejecutar hace del 100 % al 95% de lo solicitado.	Al Compilar y al ejecutar hace el 94 % al 80% de lo solicitado.	Al Compilar y al ejecutar hace el 79 % al 60% de lo solicitado.	Al Compilar y al ejecutar hace el 59 % al 50% de lo solicitado.	Al Compilar y al ejecutar hace menos del 49% de lo solicitado.	
Estructuras de control: selectivas, repetitivas Procedimientos y/o funciones (15%)	Equivalencia 20 puntos	Equivalencia 15 puntos	Equivalencia 10 puntos	Equivalencia 5 puntos	Equivalencia 0 puntos	20
	Emplea este tipo de estructuras y sentencias entre 100- 95%	Emplea este tipo de estructuras y sentencias entre 94-8 0%	Emplea este tipo de estructuras y sentencias entre 79-6 0%	Emplea este tipo de estructuras y sentencias entre 59-50%	Emplea este tipo de estructuras y sentencias en al menos 49%	
Claridad, Legibilidad y eficiencia del programa (15%)	Equivalencia 20 puntos	Equivalencia 15 puntos	Equivalencia 10 puntos	Equivalencia 5 puntos	Equivalencia 0 puntos	20
	Del 100 al 95% del código es claro, eficiente y legible	Del 94 al 80% del código es claro, eficiente y legible	Del 79 al 60% del código es claro, eficiente y legible	Del 59 al 50% del código es claro, eficiente y legible	El código no es claro, ni legible	
Variables y procedimientos y/o funciones con nombres significativos (10%)	Equivalencia 20 puntos	Equivalencia 15 puntos	Equivalencia 10 puntos	Equivalencia 5 puntos	Equivalencia 0 puntos	20
	Nombra correctamente todas las variables y procedimientos y/o funciones	Nombra correctamente la mayoría de las variables y procedimientos y/o funciones	Nombra correctamente solo algunos de las variables y procedimientos y/o funciones	Nombra correctamente muy poco a las variables y procedimientos y/o funciones	No nombra correctamente ninguna variable y a ninguna estructura	
Documentación Y Recomendaciones (10%)	Equivalencia 20 puntos	Equivalencia 15 puntos	Equivalencia 10 puntos	Equivalencia 5 puntos	Equivalencia 0 puntos	20
	Aporta dentro del código comentarios significativos y claros y aporta las recomendaciones sugeridas	Aporta dentro del código comentarios claros y aporta las recomendaciones sugeridas	Aporta dentro del código comentarios claros y aporta las recomendaciones sugeridas	No Aporta dentro del código comentarios y aporta las recomendaciones sugeridas	No Aporta dentro del código comentarios ni las recomendaciones sugeridas	

Submódulo 2 Programación en Java

Propósito del Módulo

Utilizar los lenguajes de programación en forma responsable, mediante software de programación de alto nivel, para plantear sistemas de información y desarrollar soluciones en forma sistematizada a diferentes problemáticas, favoreciendo el pensamiento algorítmico y creativo en diferentes ámbitos y contextos.

Sugerencia de Evidencia de Evaluación
• Integra Java por medio de instrucciones y estructuras para el desarrollo de programas, favoreciendo en todo momento el trabajo colaborativo y creativo en su quehacer cotidiano. • Plantea clases y objetos en Java, analizando y afrontando las consecuencias que se deriven, con la finalidad de desarrollar un trabajo metódico y organizado en las actividades que realiza en su vida cotidiana.

DOSIFICACIÓN PROGRAMÁTICA

Formación Laboral Básica: Desarrollo de Software
Módulo I: Programación Básica
Submódulo II. Programación en Java Clave: C3PJ
Semestre: 3ero. **Turno:** Matutino/Vespertino **Periodo:** 2026 – 2027A

Submódulo	Momento	Tiempo (minutos)	Conocimientos	Semana	Fecha inicio	Observaciones
SUBMÓDULO II. PROGRAMACIÓN EN JAVA	Apertura	50 min 50 min 250 min	Bienvenida. Encuadre. Reglamento. Instrumentos de evaluación. Evaluación diagnóstica. **Unidad 1.-** Introducción a Java. 1.1.- Máquina virtual de Java. 1.2.- Kit de desarrollo y entorno de ejecución (JDK)	**8**	19 – 23 de Octubre de 2026	Evaluación Extraordinaria Intrasemestral.
	Desarrollo	350 min	**Unidad 2.-** Instalación del editor y compilador de Java. 2.1.- Entorno, codificación y compilación.	**9**	26 – 30 de Octubre de 2026	30 de octubre. 1ra reunión de trabajos colegiados.
		350 min	**Unidad 3.-** Instrucciones en Java para aplicaciones básicas. 3.1.- Instrucciones de estructura:	**10**	02 – 06 de Noviembre de 2026	
		350 min	Instrucciones de estructura: 3.1.1.- Secuencial.	**11**	09 – 13 de Noviembre de 2026	
		350 min	Instrucciones de estructura: 3.1.2.-Condicional o decisión.	**12**	16 – 20 de Noviembre de 2026	17 de noviembre suspensión
		350 min	Instrucciones de estructura: 3.1.3.- Cíclicas o repetitivas.	**13**	23 – 27 de Noviembre de 2026	
		350 min	**Unidad 4.-** Clases y objetos	**14**	30 de Noviembre - 04 de Diciembre de 2026	
		350 min	**Unidad 4.-** Clases y objetos	**15**	07 - 12 de Diciembre de 2026	Revisión de portafolios y evaluación sumativa.

Submódulo	Momento	Tiempo (minutos)	Conocimientos	Semana	Fecha inicio	Observaciones
SUBMÓDULO II. PROGRAMACIÓN EN JAVA	Cierre	350 min	Situación didáctica: Reduce tu huella.	**16**	06 - 08 de Enero de 2027	 17 de diciembre 2da reunión de trabajos colegiados. 18 y 19 de diciembre Reforzamiento académico.
		150 min	Reforzamiento académico.	**17**	11 - 15 de Enero de 2027	Reforzamiento académico. 07 al 09 de enero Evaluación final.
				18	11 - 15 de Enero de 2027	Reforzamiento académico.
				19	18 - 22 de Enero de 2027	Evaluación Extraordinaria Intersemestral.
				20	25 - 29 de Enero de 2027	Jornada de capacitación
				21 y 22		Fin de periodo Receso escolar
Total: Tiempo en minutos sumando los dos submódulos		5,600 min				

Encuadre de la materia

Criterios de evaluación

Situación didáctica.	Puntaje
Reduce tu huella	50%

Actividades	Puntaje
Actividad 01. Infografía Lenguaje Java	4%
Actividad 02. Cuadro comparativo "Compiladores de Java".	4%
Actividad 03. Interfaz de Eclipse.	4%
Actividad 04. Crucigrama "Instrucciones en Java para Aplicaciones Básicas".	4%
Actividad 05. Ordena el código "Feliz Día"	4%
Total	20%

Prácticas	Puntaje
Práctica 01. Clases y Objetos "Registra tu asistencia".	5%
Practica 02. Estructura Secuencial "Promedio de tres calificaciones".	5%
Práctica 03. Estructura Condicional "Determinar de tres números dados cual es el mayor, cual el menor y realizar el producto de éstos".	5%
Práctica 04. Estructura Cíclica For "Calcula la función $e^x - x$, considerando los valores de x en el intervalo cerrado de -5 a 5".	5%
Práctica 05. Estructura Cíclica While "Programa que solicite un número cualquiera, e imprima la siguiente serie $\frac{1}{5}, \frac{3}{10}, \frac{9}{20}, \frac{27}{40}, \ldots$."	5%
Práctica 06. Estructura Cíclica Do … While "Determinar en un conjunto de n números naturales".	5%
Total	30%
Total	100%

SITUACIÓN DIDÁCTICA

TÍTULO	**REDUCE TU HUELLA**
CONTEXTO:	"La huella de carbono representa el volumen total de gases de efecto invernadero (GEI) que producen las actividades económicas y cotidianas del ser humano. Conocer el dato (expresado en toneladas de CO2 emitidas) es importante para tomar medidas y poner en marcha las iniciativas necesarias para reducirlas al máximo, empezando por cada uno de nosotros en nuestro día a día. Cada vez que viajamos en coche, cargamos el teléfono móvil o ponemos a trabajar una lavadora, entre otros miles de rutinas, dejamos atrás una estela de gases que se acumulan en la atmósfera y sobrecalientan el planeta. Estas emisiones aceleran el cambio climático, como advierte la Organización de las Naciones Unidas (ONU) en sus Objetivos de Desarrollo Sostenible (ODS), y si no las neutralizamos a tiempo con la descarbonización de la economía y otras medidas, como los impuestos ambientales, nos espera un mundo más inhóspito a la vuelta de la esquina." (fuente:https://www.iberdrola.com/sostenibilidad/huella-de-carbono) Los alumnos de tercer semestre del Colegio de Bachilleres de Tabasco, preocupados por el cuidado del medio ambiente, y para aportar una herramienta útil al PLAN ESTRATEGICO DE SOSTENIBILIDAD COBATAB, desean desarrollar herramientas de software que ayuden a medir, concientizar y tomar acciones para reducir el impacto que las actividades cotidianas de los alumnos y sus familias tienen sobre el cambio climático.

	Tomando en cuenta que la huella de carbono puede cuantificarse a nivel personal, observan que en internet hay diversas calculadoras de huellas de carbono, pero están programadas con factores de emisión globalizadas. Es por eso que los alumnos del COBATAB desarrollarán un software que permita calcular el valor aproximado de la huella de carbono que genera una familia de Tabasco, con los parámetros de factores de emisión oficiales publicados por las autoridades de México. En equipos de 6 integrantes, los estudiantes desarrollarán un programa en JAVA que tenga como objetivo interactuar con el usuario para recopilar datos de sus actividades diarias, para cuantificar el consumo de gasolina, gas y electricidad en su hogar; calcular la cantidad aproximada e CO2 expresada en Kilogramos (KgCO2e), utilizando los parámetros de factores de emisión publicados por las dependencias oficiales de México. El resultado se obtiene multiplicando el dato de consumo de energía (dato de actividad) por su correspondiente factor de emisión. Posteriormente mostrar el resultado junto con las mejores recomendaciones para reducir la huella de carbono.
PROPÓSITO	Desarrollar un programa en JAVA empleando un IDE para su compilación y ejecución, dentro de un equipo de 6 integrantes; que permita calcular la cantidad aproximada e CO2 expresada en Kilogramos (KgCO2e) recopilando datos de sus actividades diarias, utilizando los parámetros de factores de emisión publicados por las dependencias oficiales de México y posteriormente mostrar el resultado junto con las mejores recomendaciones para reducir la huella de carbono en el aula o en un espacio virtual.
CONFLICTO COGNITIVO	1. ¿De qué manera se pueden relacionar el desarrollo de software con los modelos matemáticos y los fenómenos del cambio climático? 2. ¿Qué se requiere para generar una propuesta de software para calcular la huella de CO2 de una familia o persona?

	3. ¿Qué recomendaciones darías para la reducción de emisiones y la eficiencia en el uso de recursos? 4. ¿Cuáles son las diferencias entre la programación en C++ y Java? 5. ¿Cuáles son las palabras reservadas y estructura en Java? 6. ¿Cuál es la diferencia entre clases y objetos? 7. ¿En qué dispositivos se ha utilizado Java?

Situación Didáctica: Reduce Tu Huella

Criterios de evaluación	RÚBRICA					
	Competencia Laboral Básica 2	Equipo:	**3er semestre Grupo:**	Docente:	**Estudiantes:**	
	Descriptores					
Criterio	Excelente 100-95	Muy bueno 94-80	Aceptable 79-60	Suficiente 59-50	Insuficiente 40 o menos	PTO 50
ESTRUCTURAS DE CONTROL Hace uso de las estructuras de control presentando instrucciones ordenadas. 40% 2.0 pts	Equivalencia 20 puntos	Equivalencia 15 puntos	Equivalencia 10 puntos	Equivalencia 5 puntos	Equivalencia 0 puntos	20
	Presenta el código fuente en lenguaje JAVA, con las instrucciones organizadas y categorizadas, en forma física o digital además incluye comentarios, presenta todas las estructuras de control.	Presenta el código fuente en lenguaje JAVA, con las instrucciones organizadas y categorizadas, en forma física o digital además incluye comentarios, presenta al menos dos estructuras de control.	Presenta el código fuente en lenguaje JAVA, con las instrucciones organizadas y categorizadas, en forma física o digital además incluye comentarios, todo lo presenta la estructura de secuencial.	Presenta el código fuente en lenguaje JAVA, con las instrucciones organizadas y categorizadas, en forma física o digital.	Entrega un código incompleto en JAVA de manera física o digital.	
Presentación de clases y objetos lógicamente expresados. 20% 1.0 pt	Equivalencia 10 puntos	Equivalencia 7.5 puntos	Equivalencia 5 puntos	Equivalencia 2.5 puntos	Equivalencia 0 puntos	10
	Presenta las clases y los objetos debidamente ordenados y una sintaxis correcta verificando que el programa se ejecuta sin errores.	Presenta las clases y los objetos debidamente ordenados y una sintaxis correcta sin verificar que el programa se ejecuta sin errores.	Presenta las clases y los objetos debidamente ordenados y con detalles en la sintaxis.	Presenta las clases y los objetos desordenados y con detalles en la sintaxis.	Carece de las clases y los objetos, y el código que presenta contiene errores.	

	Equivalencia 10 puntos	Equivalencia 7.5 puntos	Equivalencia 5 puntos	Equivalencia 2.5 puntos	Equivalencia 0 puntos	10
Utiliza funciones. 20% 1.0 pt	Presenta sin errores el uso de funciones que cuentan con las siguientes características: -Uso de parámetros -Retorno de valores -Recursividad.	Presenta sin errores, el uso de funciones que cuenten con las siguientes características: -Uso de parámetros -Retorno de valores.	Presenta errores en el uso de funciones, cuentan con las siguientes características: -Uso de parámetros.	Presenta errores en el uso de funciones, sin el uso de parámetros.	Presenta funciones con errores. El código no es claro, ni legible.	
	Equivalencia 5 puntos	Equivalencia 3.75 puntos	Equivalencia 2.5 puntos	Equivalencia 1.25 puntos	Equivalencia 0 puntos	5
Trabajo colaborativo. 10% 0.5 pts	Se asignaron funciones en el equipo y todos contribuyeron en la corrección y elaboración del software, mostrando interés y orden en la actividad.	Se asignaron funciones en el equipo y todos contribuyeron en la corrección del software, pero mostraron desinterés y desorden.	Se asignaron funciones en el equipo, pero algunos participantes no contribuyeron en la corrección del software, mostrando apatía y desorden.	Se asignaron funciones en el equipo, pero los integrantes no contribuyeron en la corrección del software, demostrando desinterés.	No se asignaron funciones en el equipo. El software fue corregido y presentado por un único participante.	
	Equivalencia 5 puntos	Equivalencia 3.75 puntos	Equivalencia 2.5 puntos	Equivalencia 1.25 puntos	Equivalencia 0 puntos	5
Exposición. 10% 0.5 pts	Todos los integrantes del equipo exponen con claridad y fluidez el software, mencionando el uso de funciones, clases y objetos, utilizando el vocabulario correcto del lenguaje JAVA.	Todos los integrantes del equipo exponen con claridad y fluidez el software, mencionando el uso de funciones, clases y objetos, no utilizan el vocabulario correcto del lenguaje JAVA.	Todos los integrantes del equipo exponen con claridad y fluidez el software, mencionan con dificultad el uso de funciones, clases y objetos, no utilizan el vocabulario correcto del lenguaje JAVA.	No todos los integrantes del equipo exponen y quienes lo hacen, con dificultad expresan el uso de funciones, clases y objetos, no utilizan el vocabulario correcto del lenguaje JAVA.	Algunos integrantes del equipo exponen el software, mencionan con dificultad el uso de funciones, clases y objetos, no utilizan el vocabulario correcto del lenguaje JAVA.	

Evaluación Diagnóstica Submódulo 2

PROGRAMACIÓN EN JAVA

NOMBRE_____**GRUPO:** ____

INSTRUCCIONES: LEE Y SUBRAYA LA RESPUESTA CORRECTA.

1.- ¿Qué es java?

A) Java es un lenguaje de programación.
C) Una aplicación.
B) Un código.
D) Un archivo.

2.- ¿Cuál es el recolector de basura que utiliza Java?

A) Fill
C) Garbage collector
B) Papelera
D) Core

3.- ¿Cuál es la función de garbage collector?

A) Recolectar información
C) Ejecuta
B) Detectar cuando una variable no va a ser utilizada de nuevo
D) Modifica

4.- En java, existen dos conjuntos de herramientas que nos permiten trabajar, ¿cuáles son?

A) Programar y procesar
B) Mostrar el texto y definir color de texto
C) Crear métodos y ejecutar programas
D) El JDK (herramientas de desarrollo) y el JRE (entorno de ejecución)

5.- Es un conjunto de librerías y programas que permiten el desarrollo del lenguaje, es decir, compilar, ejecutar generar documentación.

A) La herramienta de desarrollo de java
C) Un procesador
B) Un sistema operativo
D) Un software

6.- Es el que permite la ejecución de los programas. Está formado principalmente por la máquina virtual y una serie componentes necesarios para la ejecución de los programas.

A) JKE para ejecución de software
B) RJE para el software de java
C) JRE para ejecutar las aplicaciones
D) JRE para ejecutar aplicaciones

7.- Los entornos de desarrollo para java más utilizados:

A) Netbeans, Intellij y Eclipse
B) Browse
C) Java project
D) Workspace

8.- ¿Qué es android?

A) Es una plataforma
B) Un código
C) Servicio de google
D) Sistema operativo para dispositivos móviles

9.- La principal finalidad desde el punto de vista del desarrollador es:

A) Que la aplicación sea portable
B) Aplicaciones web de google
C) Obtener el máximo partido de la plataforma
D) Posicionar sus servicios de búsqueda

10.- Uno de los grandes pilares de java es:

A) Los programas funcionan en cualquier plataforma
B) Su sistema operativo
C) Otros lenguajes
D) Sus versiones específicas

11.- ¿Qué es un IDE (Integrated Development Environment)?

A) Un código
B) Un software
C) Un servicio
D) Básicamente es un compilador

12.- ¿Qué se necesita para llevar a cabo la instalación de java?

A) Necesitamos un IDE (Entorno de Desarrollo Integrado)

B) Internet

C) Un archivo

D) Un código

13.- ¿Qué significa JDK?

A) Java development kit

B) Herramientas

C) Java

D) Crear programas

Lectura No. 1 "Introducción al Lenguaje de Programación Java"

Instrucciones: *Realiza la Lectura 1. "Introducción al Lenguaje de Programación en Java" subrayando las ideas principales, y al finalizar realiza la Actividad 1 – Infografía lenguaje Java*

LECTURA 1. "INTRODUCCIÓN AL LENGUAJE DE PROGRAMACIÓN EN JAVA"

Java no fue creado originalmente para la red internet. Sun Microsystems comenzó a desarrollarlo con el objetivo de crear un lenguaje, independiente de la plataforma y del sistema operativo, para el desarrollo de electrónica de consumo (dispositivos electrónicos inteligentes, como televisores, vídeos, equipos de música, etc.). El proyecto original, denominado **Green** comenzó apoyándose en C++, pero a medida que se progresaba en su desarrollo el equipo creador, comenzó a encontrarse con dificultades, especialmente de portabilidad. Para evitar estas dificultades, decidieron desarrollar su propio lenguaje y en agosto de 1991 nació un nuevo **lenguaje orientado a objetos**. Con el nombre de Oak. A mitad de 1993, se lanzó Mosaic, el primer navegador para la Web y comenzó a crecer el interés por Internet (y en particular por la World Wide Web).

Entonces, se rediseñó el lenguaje para desarrollar aplicaciones para internet y, en enero de 1995, Oak se convirtió en Java. Sun lanzó el entorno **JDK 1.0** en 1996, primera versión del kit de desarrollo de dominio público, que se convirtió en la primera especificación formal de la plataforma Java. Aunque la primera comercial se denominó JDK 1.1 en 1997. En diciembre de 1998 Sun lanzó la plataforma Java 2 (conocida como JDK 1.2 durante su fase de pruebas beta). Conocida como Java 2 JDK, versión 1.3. Los programas Java se pueden incluir ((embeber)) o ((empotrar)) en páginas HTML y descargarse por navegadores Web para llevar animaciones e interacciones a los clientes Web. La potencia de Java no se limita a aplicaciones Web; es un lenguaje de programación de propósito general que posee características completas para programación de aplicaciones independientes o autónomas.

Java, como lenguaje, es fundamentalmente orientado a objetos. Se diseñó desde sus orígenes como verdadero lenguaje orientado a objetos.

Un Poco de Historia de Java y Evolución

Java es una mezcla de los mejores elementos de los lenguajes de programación exitosos. Aunque ha sido fuertemente ligado a Internet, es importante recordar que Java es un lenguaje de programación de uso general. Las innovaciones y desarrollo de los lenguajes de programación ocurren por dos razones fundamentales:

- Para adaptarse a los cambios en ambientes y usos.
- Para implementar refinamientos y mejoras en el arte de la programación.

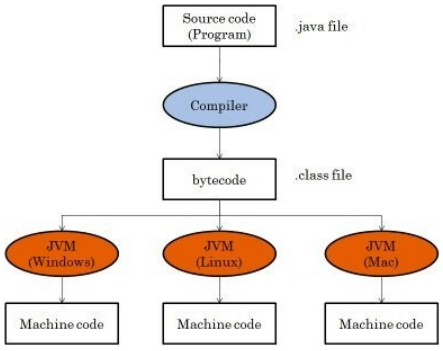
Java es un descendiente de C++ que a su vez es descendiente directo de C. Muchas características de Java se han heredado de estos dos lenguajes. **De C, Java ha heredado su sintaxis y de C++, las características fundamentales de programación orientada a objetos.** El diseño original de Java fue concebido por James Gosling, Patrick Naughton, Chris Warth, Ed Frank y Mike Sheridan, ingenieros y desarrolladores de Sun Microsystems en 1991, que tardaron 18 meses en terminar la primera versión de trabajo. Este lenguaje se llamó inicialmente «Oak», y se le cambió el nombre por Java en la primavera de 1995.

Sorprendentemente, la inquietud original para la creación de «Oak» no era Internet. En realidad, se buscaba un lenguaje independiente de la plataforma (es decir, de arquitectura neutra) que se pudiera utilizar para crear software que se incrustara en dispositivos electrónicos diversos tales como: controles remotos, automóviles u hornos de microondas.

El problema, en consecuencia, se convertía en compiladores caros y en gran consumo de tiempo para crear los programas. Sobre esas premisas, Gosling y sus colegas comenzaron a pensar en un lenguaje portable, independiente de la plataforma que se pudiera utilizar para producir código que se ejecutara en una amplia variedad de CPU y bajo diferentes entornos. Entonces comenzó a aparecer el nuevo proyecto y se decidió llamarle Java.

El bytecode

La clave que permite a Java resolver los principales problemas en la computación, el de la seguridad y el de la portabilidad, es que la salida del compilador de Java no es un código ejecutable, sino un bytecode. **El bytecode** es un conjunto de instrucciones altamente optimizado diseñado para ser ejecutado por una máquina virtual la cual es llamada Java Virtual Machine (**JVM**, por sus siglas en inglés).

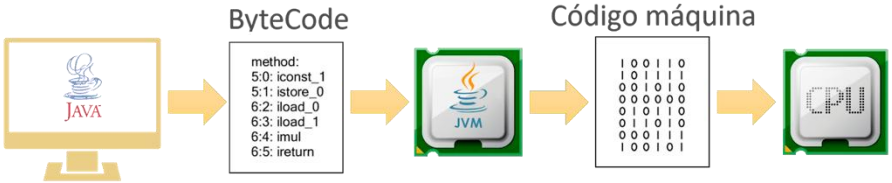


En esencia, la **máquina virtual** original fue diseñada como un intérprete de bytecode. Esto puede resultar un poco sorprendente dado que muchos lenguajes de programación modernos están diseñados para ser compilados en código ejecutable pensando en lograr el mejor rendimiento.

Traducir un programa Java en bytecode hace que su ejecución en una gran variedad de entornos resulte mucho más sencilla, y la razón es que, para cada plataforma, sólo es necesario implementar el intérprete de Java. Una vez que el sistema de ejecución existe para un ambiente determinado, cualquier programa de Java puede ejecutarse en esa plataforma. Además, la ejecución del bytecode a través de la JVM es la manera más fácil de crear código auténticamente portable. El hecho de que Java sea interpretado también ayuda a hacerlo seguro. Como la ejecución de cada programa de Java está bajo el control de la JVM, ésta puede contener al programa e impedir que se generen efectos no deseados en el resto del sistema.

Java como Lenguaje y Plataforma de Programación

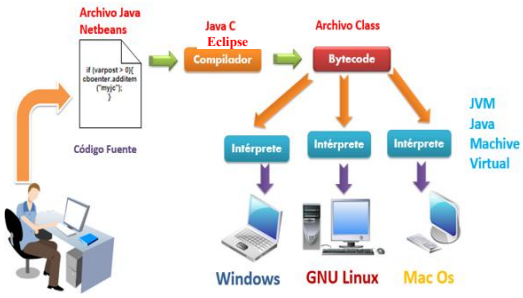
Java es un lenguaje de programación de propósito general, posiblemente, uno de los más populares y utilizados en el desarrollo de programas de software, especialmente para internet y web; actualmente se encuentra en numerosas aplicaciones, dispositivos, redes de comunicaciones, etcétera, como:



- Servidores web.
- Bases de datos relacionales.
- Sistemas de información geográfica (SIG/GIS, Geographical Information System).
- Teléfonos celulares (móviles).
- Sistemas de teledetección.
- Asistentes digitales personales (PDA).
- Sistemas medioambientales.

La Máquina Virtual de Java (JVM, Java Virtual Machine)

La máquina virtual de Java se denomina al procesador o **entorno virtual** que se utiliza para interpretar los bytecodes de los binarios de Java, ya que como sabemos Java se hizo para correr en cualquier plataforma sin recompilar los binarios. De esta manera este entorno virtual se puede obtener para nuestra arquitectura y sistema operativo sin modificaciones a nuestro programa original (esto no es cierto si utilizamos una mala dinámica de programación). Podemos entonces generar un binario y este podrá Correr en Linux, MAC OSX, FreeBSD, Solaris, o Windows, y para las arquitecturas disponibles en las que podamos obtener la JVM, como ser AMD64, SPARC, PIV, etc.

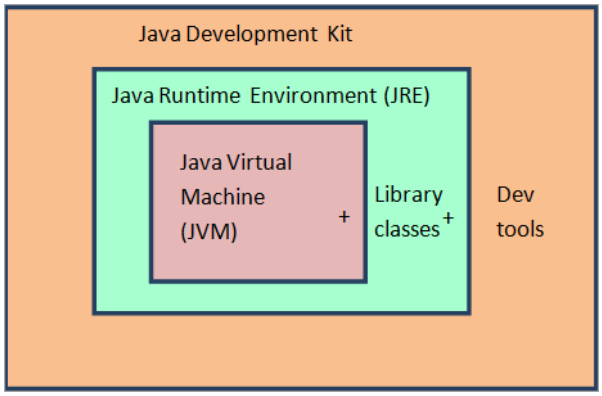


La máquina virtual de Java ha tenido la característica de ser un entorno de ejecución pesado en términos de recursos del procesador y memoria, que por medio de una administración rigurosa del sistema operativo estos podrían llegar a ser insuficientes y las aplicaciones ejecutarse de manera muy lenta. Esto no es cierto en la actualidad, existen alternativas a la JVM provista por Sun Microsystems que permiten una velocidad comparable a una aplicación compilada en C++ nativa en la arquitectura, un ejemplo de esto es Kaffe. Kaffe (www.kaffe.org) es una máquina de Java OpenSource que puede compilarse sin mayores modificaciones en nuestra arquitectura necesaria y correrá increíblemente más rápida que la distribución estándar de JVM de Sun Microsystems y consumirá muchos menos recursos.

Kit de desarrollo y Entorno de ejecución (JDK, JRE)

El Kit de desarrollo conocido como **JDK** (Java Development Kit) provee de un compilador, un mecanismo para comprimir un proyecto en un solo archivo de tipo **JAR** (que es compatible con ZIP) y un entorno de ejecución para nuestros binarios.

Este **JRE** (Java Runtime Environment) también puede redistribuirse sin problemas de licencias. En las plataformas basadas en Linux, existen mejores herramientas de desarrollo y por supuesto en casi todas las distribuciones de Linux, se encuentra disponible Kit de desarrollo o JDK.



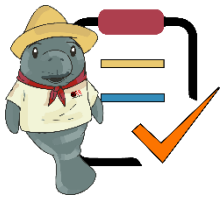
BIBLIOGRAFÍA DE LA LECTURA 1

Java 2 Manual de Programación, Luis Joyanes Aguilar, Matilde Fernández, Osborne McGrawn-Hill, Salamanca, España, pág. XIV.

Programación en Java 6, algoritmos y programación orientada a objetos. Luis Joyanes Aguilar, Ignacio Zahonero Martínez, McGrawn-Hill, 2011, México D.F., pág. 20.

Aprendiendo Java y Programación Orientada a Objetos. Gustavo Guillermo Pérez. 2008.

Manual de Referencia Java, Herbert Schildt, McGrawHill, 2009, Ciudad de México, pág. 9.

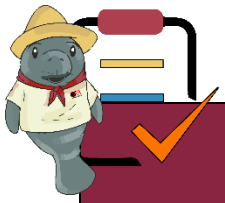


Actividad 01 "Infografía Lenguaje Java"

Instrucciones: Realiza una infografía donde muestras los aspectos que lograste comprender de la lectura 1. Considera como apoyo los siguientes puntos.

1) ¿Qué es Java? 2) ¿Qué es un Bytecode? 3) ¿Qué significa JVM?

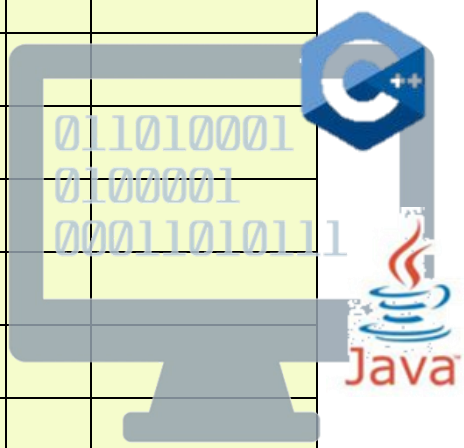
4) Por qué usar JDK. 5) En qué objetos de nuestra cotidiana se aplica la programación Java.

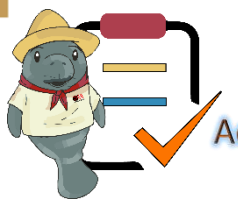


INSTRUMENTO DE EVALUACIÓN
LISTA DE COTEJO
Actividad 01. "Infografía Lenguaje Java"

DATOS GENERALES	
Nombre(s) del alumno(s)	**Matricula(s)**
Producto: Infografía Lenguaje Java	**Fecha**
Submódulo II: Programación en Java	**Periodo: 2026 – 2027A**
Nombre del docente	**Firma del docente**

CRITERIO	INDICADORES	VALOR OBTENIDO		PONDERACIÓN	CALIF	OBSERVACIONES Y/O SUGERENCIAS DE MEJORA
		SÍ	NO			
1	Entrego en tiempo y Forma.			**2**		
2	Agrego la información correcta.			**2**		
3	Se aprecia el conocimiento obtenido del tema.			**1**		
4	Comprende los conceptos de Java para su utilización.			**2**		
5	Utiliza la tecnología para desarrollar la actividad.			**2**		
6	Mantiene interés en la comprensión de la actividad.			**1**		
				CALIFICACIÓN		





Actividad 02 Cuadro Comparativo "Compiladores de Java"

Instrucciones: Después de la presentación dada por el docente sobre Entornos de Desarrollo Integrados (IDE).

En parejas, investiguen 5 compiladores (IDE) de Java y elabora una tabla comparativa donde muestres la imagen, nombre, sistemas operativos en los que se puede instalar, lenguajes adicionales que permite trabajar y una descripción breve donde presentas tus comparaciones y observaciones.

IMAGEN	NOMBRE	SISTEMAS OPERATIVOS	LENGUAJES	OBSERVACIONES
1.				
2.				
3.				
4.				
5.				

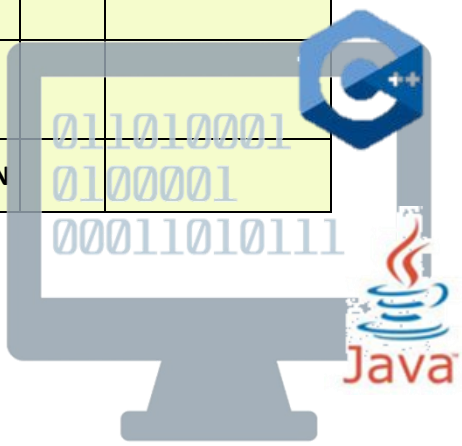
CONCLUSIONES:

Dirección Web
https://www.conmasfuturo.com/los-entornos-de-desarrollo-java-los-mejores-entornos-de-desarrollo-java-para-ninos-y-adolescentes/

INSTRUMENTO DE EVALUACIÓN
LISTA DE COTEJO
Actividad 02. "Cuadro Comparativo "Compiladores de Java"

DATOS GENERALES	
Nombre(s) del alumno(s)	**Matricula(s)**
Producto: Cuadro Comparativo "Compiladores de Java"	**Fecha**
Submódulo II: Programación en Java	**Periodo: 2026 – 2027A**
Nombre del docente	**Firma del docente**

CRITERIO	INDICADORES	VALOR OBTENIDO		PONDERACIÓN	CALIF	OBSERVACIONES Y/O SUGERENCIAS DE MEJORA
		SÍ	NO			
1	Aporta ideas que provienen de fuentes confiables.			**1**		
2	Exponen sus conclusiones con claridad y fluidez			**2**		
3	Presentan en la tabla **Imagen, Nombre, Sistemas Operativos, Lenguajes y Observaciones.**			**3**		
4	Utiliza las tecnologías de información y comunicación para obtener información o desarrollar la tabla.			**2**		
5	Actúan con tolerancia ante las opiniones de los demás.			**1**		
6	Muestra sus opiniones o comparaciones, sustentando su postura en las observaciones.			**1**		
	CALIFICACIÓN					



Lectura No. 2 "Instalación de herramientas para programar con Java"

Instrucciones: *Realiza la Lectura 2. "Instalación de herramientas para programar con Java" subrayando las ideas principales, y al finalizar realiza la Actividad 3 – Interfaz gráfica de Eclipse*

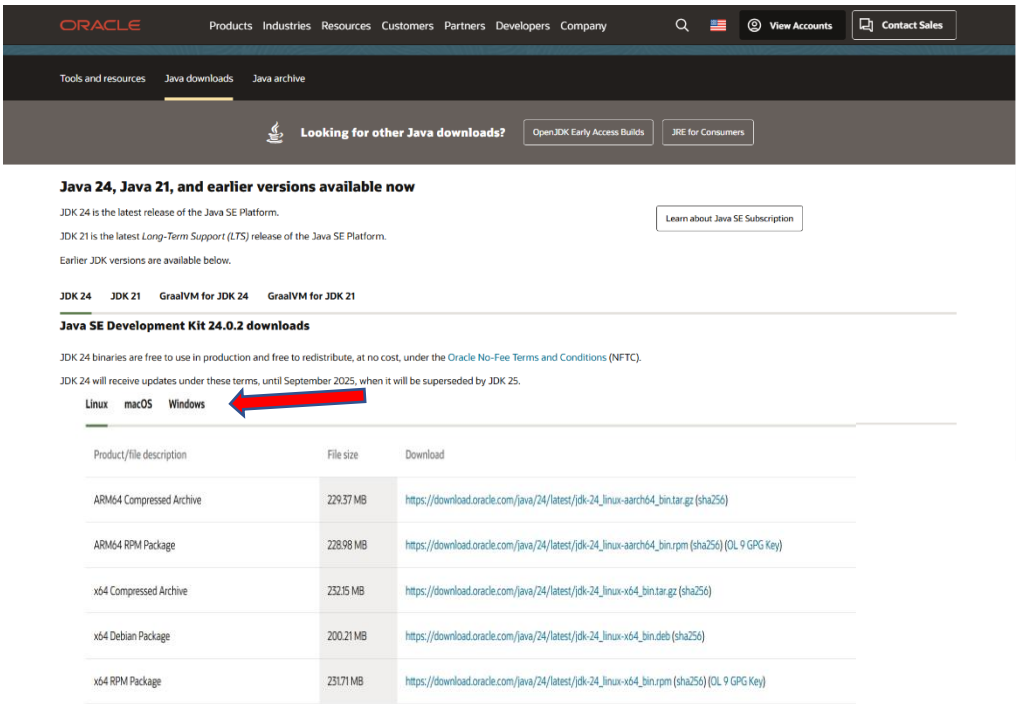
LECTURA 2. "INSTALACIÓN DE HERRAMIENTAS PARA PROGRAMAR CON JAVA"

1. Como primer punto debemos tener instalado el kit de Desarrollo Java o JDK, el cual contiene el conjunto de librerías o bibliotecas de Java.

Para ello deberás ir a la siguiente dirección:

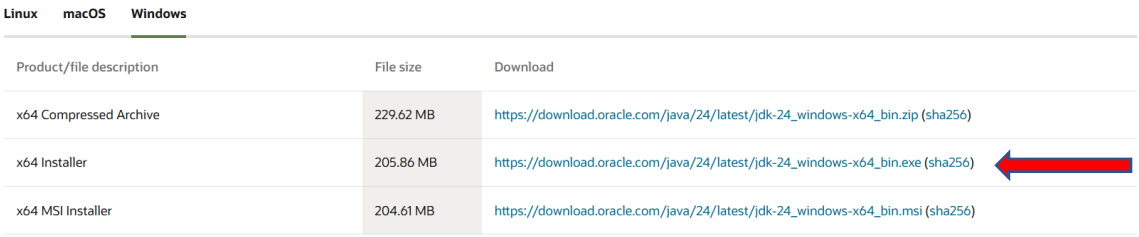
https://www.oracle.com/java/technologies/downloads/?er=221886

Te enviará a la siguiente ventana en donde dará un clic sobre JDK download

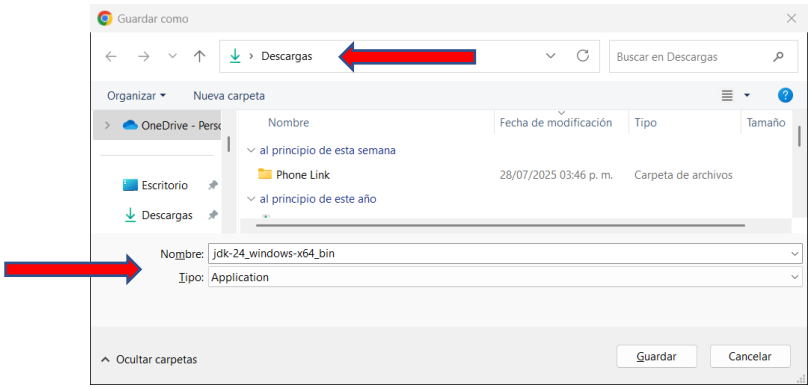


Te llevara a esta nueva pantalla

Te deslizaras en esta pantalla hasta llegar a esta. En donde te ubicaras hasta Windows x64 Installer y pulsaras en la opción de lado derecho https://download.oracle.com/java/24/latest/jdk-24_windows-x64_bin.exe



Para que inicie la descarga. Antes de eso deberás aceptar el acuerdo de licencia de Oracle. Seleccionaras en el cuadro de opción y pulsaras sobre para que se active en color verde el enlace de Download https://download.oracle.com/java/24/latest/jdk-24_windows-x64_bin.exe, paso seguido darás clic sobre este para empezar la descarga de JDK.



Ya que hayas descargado el JDK, solo dale clic al archivo que descarga e inicia el proceso de instalación dando clic al botón de next solamente (posteriormente continúa dando clic en este mismo, hasta que se termine de realizar por completo la instalación del JDK).

Puedes apoyarte con el siguiente enlace en caso de no poder hacer todo el proceso.

https://youtu.be/hCBEavs08as

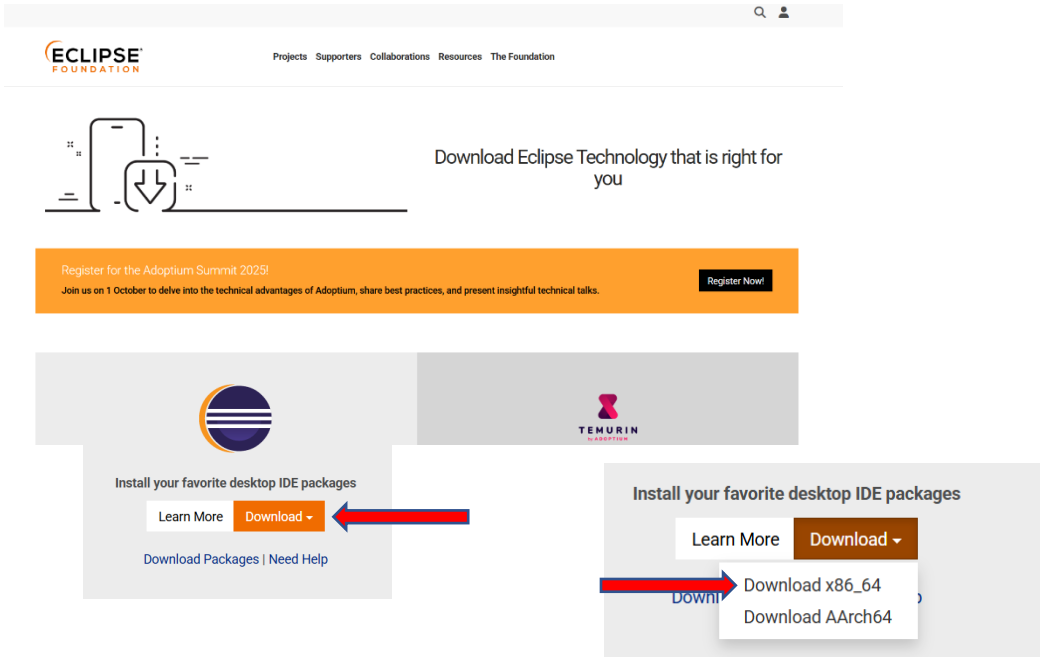
Instalar el IDE o Entorno de Desarrollo Integrado (Significado en español). El cual servirá para compilar, y ejecutar los códigos de programación de este lenguaje, existen varios IDE en este curso utilizaremos uno de los más utilizados por los programadores o desarrolladores en Java conocido como Eclipse, el cual tiene muy buenas características y se complementa con una gran cantidad de plugins (complementos) que ayudan bastante a la hora del desarrollo o programación.

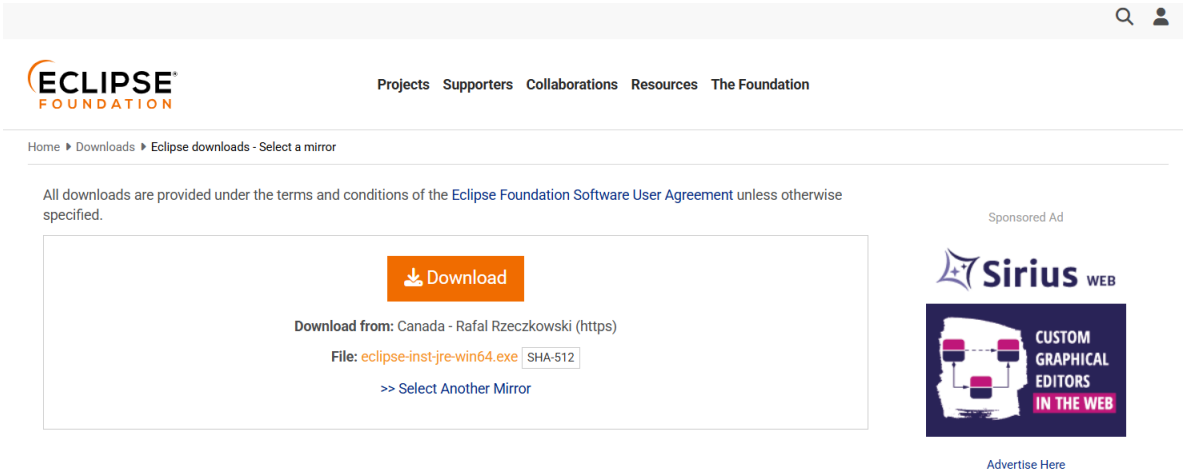
Para poder descargar este IDE deberás dar clic en el siguiente enlace:

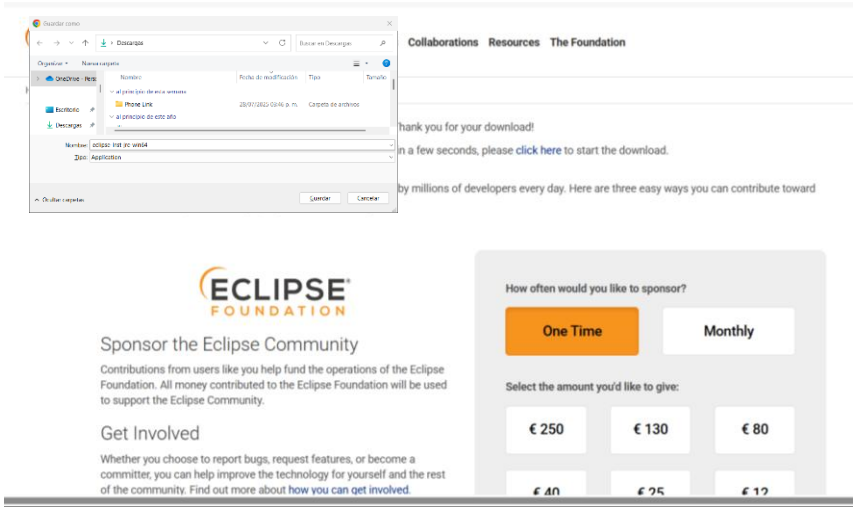
https://www.eclipse.org/downloads/

A continuación, se visualizará la siguiente ventana. Y darás clic sobre el botón que dice Download x86_64.

La cual te llevara a esta nueva ventana. En donde darás clic sobre el enlace File: eclipse-inst-jre-win64.exe, que te llevara a la descarga del archivo instalador de eclipse.







Una vez descargado, tendrás un archivo comprimido que, al descomprimir, te creará una carpeta llamada **eclipse**, que puedes localizar donde queramos. Dentro de esta carpeta, hay un conjunto de archivos y carpetas, entre los cuales se encontrará el ejecutable. Si lo ejecutas por primera vez, tras el logo inicial, te aparecerá una ventana donde te pedirá que indiques el **workspace**, que no es más que la carpeta donde se guardarán, en principio, tus proyectos. Por defecto deja la que te pone y no selecciones otra.

Puedes apoyarte con el siguiente enlace, en caso de no poder realizar todo el procedimiento de instalación.

https://youtu.be/MYbyzNWnrzU

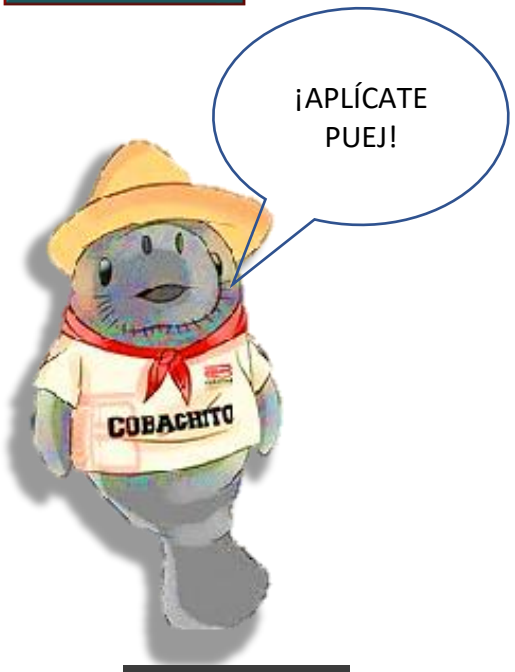
"Instalación del IDE de Java"

Compilador para PC

Da clic en el link o escanea el código Qr.

Descarga Eclipse.
https://www.eclipse.org/downloads/

Instalación Eclipse.
https://youtu.be/MYbyzNWnrzU



Compilador Web

https://www.jdoodle.com/online-java-compiler

Online Java Compiler | Java Editor

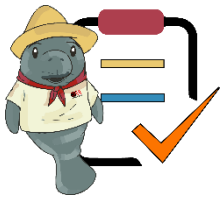
Compiladores para Móvil

JStudio - ide for java

Jvdroid - IDE for Java

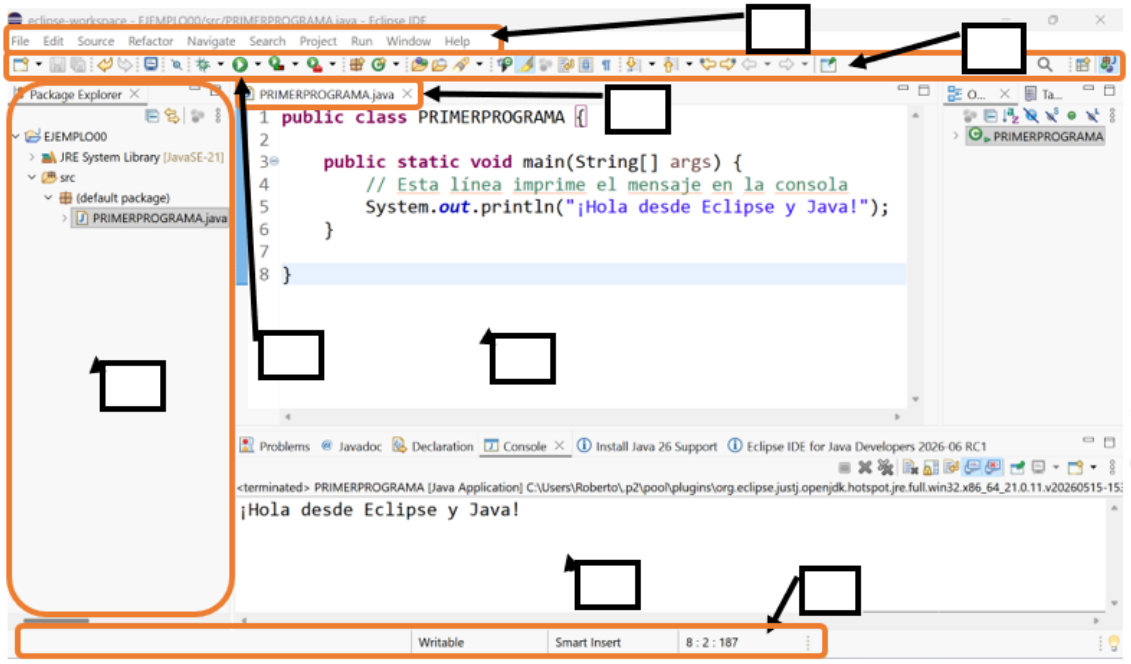
CodeBrew - IDE for Java

Java Compiler IDE



Actividad 03 Interfaz Gráfica de Eclipse

Instrucciones: Coloca en el recuadro el número que le corresponda al elemento de la ventana, tomando en cuenta los datos que se presentan en el listado de abajo.

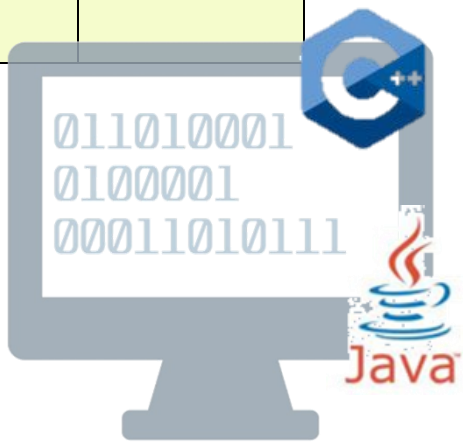


1. Barra de menú.
2. Barra de herramientas
3. Nombre del programa/Código.
4. Línea de estado.
5. Icono de compilar y ejecutar.
6. Área de edición de código.
7. Consola.
8. Explorador de paquetes.

INSTRUMENTO DE EVALUACIÓN
LISTA DE COTEJO
Actividad 03. Interfaz Gráfica de Eclipse

DATOS GENERALES	
Nombre(s) del alumno(s)	**Matricula(s)**
Producto: Interfaz Gráfica de Eclipse	**Fecha**
Submódulo II: Programación en Java	**Periodo: 2026 – 2027A**
Nombre del docente	**Firma del docente**

CRITERIO	INDICADORES	VALOR OBTENIDO		PONDERACIÓN	CALIF	OBSERVACIONES Y/O SUGERENCIAS DE MEJORA
		SI	NO			
1	Entrego en tiempo y Forma.			**2**		
2	Identifico correctamente la ubicación de los números.			**2**		
3	Se aprecia el conocimiento obtenido de la aplicación.			**1**		
4	Comprende de manera gráfica, el elemento que se señala en la ventana.			**2**		
5	Utiliza la tecnología para desarrollar la actividad.			**2**		
6	Mantienen interés en la comprensión de la actividad.			**1**		
	CALIFICACIÓN					



Lectura No. 3 "Instrucciones en Java para Aplicaciones Básicas"

Instrucciones: *Realiza la Lectura 3. "Instrucciones en Java para Aplicaciones Básicas" subrayando las ideas principales, y al finalizar realiza la Actividad 4 – Instrucciones en Java*

LECTURA 3. "INSTRUCCIONES EN JAVA PARA APLICACIONES BÁSICAS"

Estructura de un programa en Java

Un programa describe cómo un ordenador debe interpretar las órdenes del programador para que ejecute y realice las instrucciones dadas tal como están escritas.

El siguiente programa muestra un mensaje en la consola con el texto "Hola Mundo".

```
/**Este programa escribe el texto "Hola Mundo" en la
consola*Utilizando el método System.out.println()
*/
public class HolaMundo {
        public static void main (String[] args) {
           System.out.println("Hola Mundo");
        }
}
```

Describiendo el programa anterior:

Comentario. Este programa inicia con un comentario. El delimitador de inicio de un comentario es /* y el delimitador de fin de comentario es */. El texto que tendría el comentario completo seria: Este programa escribe el texto Hola Mundo en la consola utilizando el **método** System.out.println(). Estos ayudan a explicar aspectos relevantes de un programa y lo hacen más legible; pudiendo identificar que hace una línea de código o varias líneas, o bien un método, una función, clase, etc.

Definición de clase. La primera línea del programa, después del primer comentario, define una clase que se llama HolaMundo. La definición de la clase comienza por el carácter { y termina con el carácter }. El nombre de la clase lo define el programador.

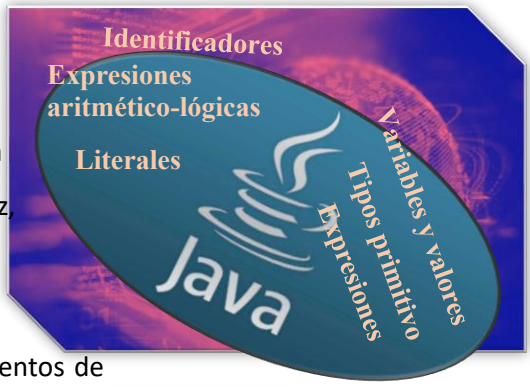
Definición de Método. Después de la definición de clase se escribe la definición del método main(). Todos los programas Java deben incluir un método main(). Este método indica las sentencias a realizar cuando se ejecuta un programa. Un método es una secuencia de sentencias ejecutables. Quedan definidas por los caracteres { y } que indican el inicio y fin del método, respectivamente.

Sentencia. Dentro del método main() se incluye una sentencia para mostrar un texto por la consola. Los textos siempre se escriben entre comillas dobles ("Hola Mundo") para diferenciarlos de otros elementos del lenguaje. Todas las sentencias de un programa en Java deben terminar con el símbolo punto y coma (;). Este símbolo indica al compilador (IDE) que ha finalizado una sentencia.

Una vez que se ha escrito el código el siguiente paso es compilarlo y ejecutarlo. Para comprobar si es correcto.

Elementos de un programa Java

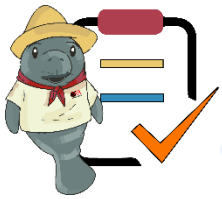
En la programación en Java, la definición **léxica** y **sintáctica** de los elementos de un programa Java: comentarios, identificadores, variables y valores, tipos primitivos, literales, operadores, expresiones y expresiones aritmético – lógicas. Mismos elementos que podemos observar en C++.



BIBLIOGRAFÍA DE LA LECTURA 3

Programación en Java 6, algoritmos y programación orientada a objetos. Luis Joyanes Aguilar, Ignacio Zahonero Martínez, McGrawn-Hill, 2011, México D.F., pág. 20.

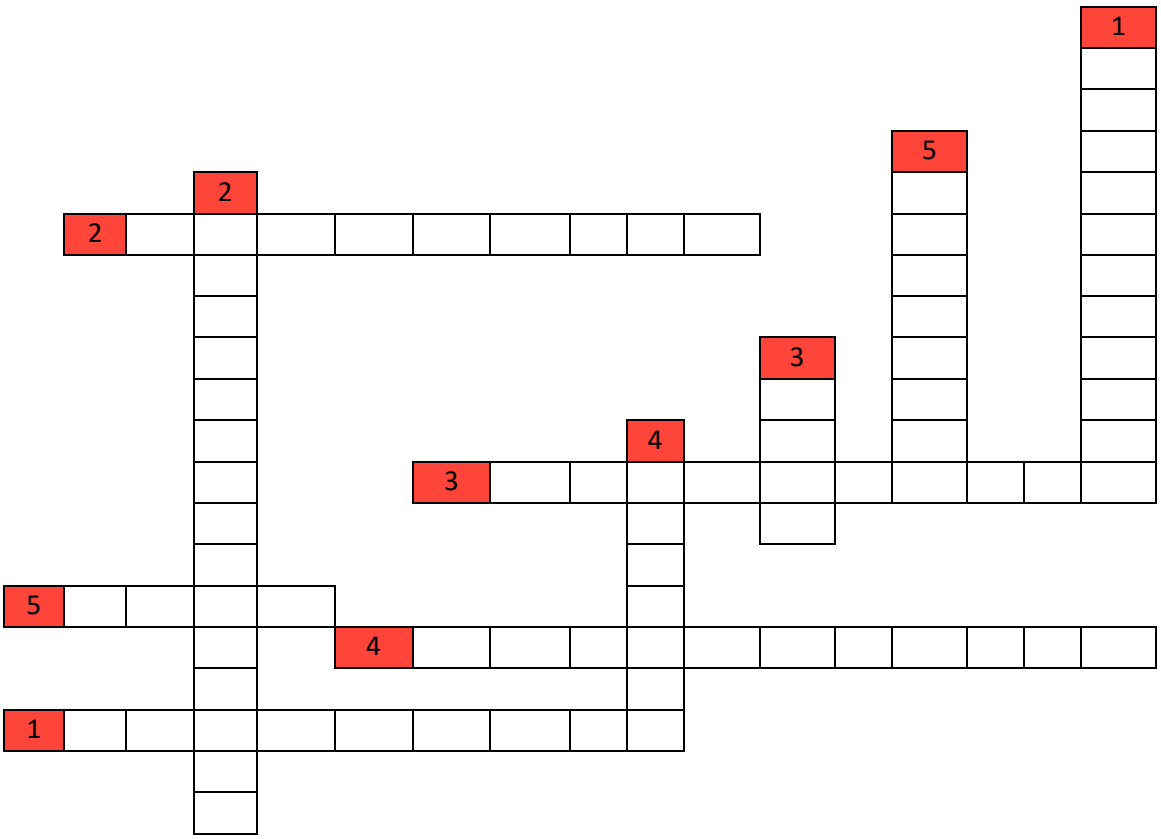
Jorge Martínez Ladron de Guevara, G- Tec. (2020). Fundamentos de programación en Java (Spanish Edition). Madrid, España: Independently published.



Actividad 04. Crucigrama

"Instrucciones en Java para Aplicaciones Básicas"

Tomando como base la lectura anterior y los elementos estudiados en el tema "Instrucciones en C++" del submódulo 1. Programación en C++, resuelve el crucigrama:



Vertical

1.- El uso de éstos, hace más claro y legible un programa. En ellos se debe decir que se hace, para que y cuál es el fin de nuestro programa. Conviene utilizarlos siempre que merezca la pena ya que hace una aclaración sobre el programa.

2.- Éstos son elementos que el programador tiene libertad para elegir el nombre de las variables, los métodos, entre otros aspectos de un programa y tienen reglas muy estrictas sobre los nombres que se utilizan para las clases, las variables o los métodos.

3.- Este tipo de dato permite representar cualquier caracter Unicode.

4.- Tipos de datos ENTEROS: byte, short, int y long

5.- Se utiliza este tipo de dato para representar los valores lógicos verdadero y falso.

Horizontal

1.- Un programa Java utiliza estos elementos para almacenar valores, realizar cálculos, modificar los valores almacenados, mostrarlos por la consola, almacenarlos en disco, enviarlos por la red, etc.

2.- Éstas son la manera en que se escriben los valores para cada uno de los tipos primitivos.

3.- Estos elementos permiten realizar cálculos o acciones con el uso de las literales.

4.- Permite realizar operaciones entre valores utilizando distintos operadores. Son útiles para representar las fórmulas matemáticas que se utilizan para realizar cálculos.

5.- Los tipos de datos de coma flotante permiten almacenar números muy grandes, muy pequeños o que contienen coma decimal.

https://es.educaplay.com/juego/9949849-instrucciones_en_java.html

INSTRUMENTO DE EVALUACIÓN
LISTA DE COTEJO
Actividad 04. Crucigrama "Instrucciones en Java para Aplicaciones Básicas"

DATOS GENERALES	
Nombre(s) del alumno(s)	**Matricula(s)**
Producto: Crucigrama "Instrucciones en Java para Aplicaciones Básicas"	**Fecha**
Submódulo II: Programación en Java	**Periodo: 2026 – 2027A**
Nombre del docente	**Firma del docente**

CRITERIO	INDICADORES	VALOR OBTENIDO		PONDERACIÓN	CALIF	OBSERVACIONES Y/O SUGERENCIAS DE MEJORA
		SÍ	NO			
1	Entrego en tiempo y Forma.			**2**		
2	Identifico correctamente las respuestas en los espacios.			**2**		
3	Se aprecia el conocimiento obtenido de la aplicación.			**1**		
4	Comprende el concepto presentado.			**2**		
5	Utiliza la tecnología para desarrollar la actividad.			**2**		
6	Mantienen interés en la comprensión de la actividad.			**1**		
				CALIFICACIÓN		

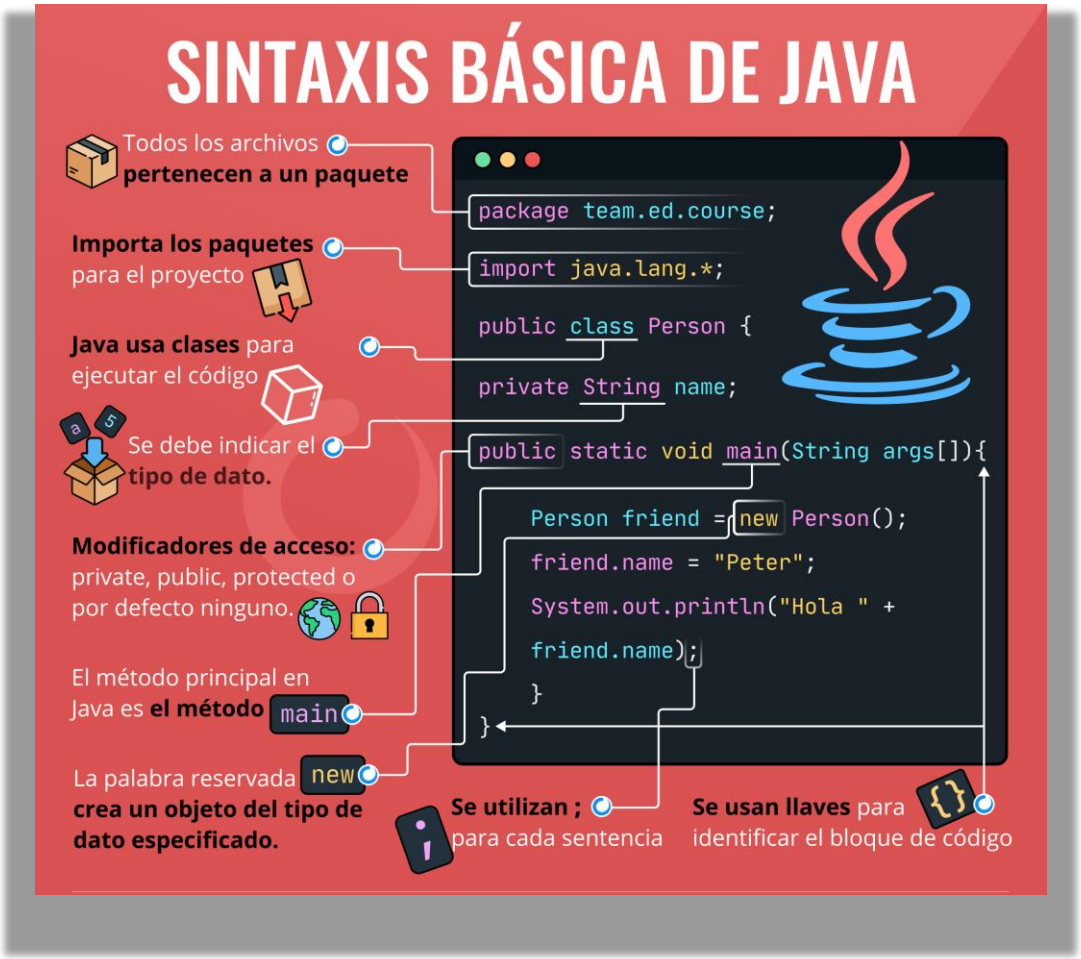
Lectura No. 4 "Estructuras de Java"

Instrucciones: *Realiza la Lectura 4. "Estructuras de Java" subrayando las ideas principales, y al finalizar realiza la Actividad 5 – Ordena el código "Feliz día"*

LECTURA 4. "ESTRUCTURAS DE JAVA"

Estructura de un programa en Java

Java es uno de los lenguajes de programación con mayor demanda laboral en grandes empresas de todo el mundo. ¡Tienes que aprenderlo!



Entrada y lectura de datos en Java

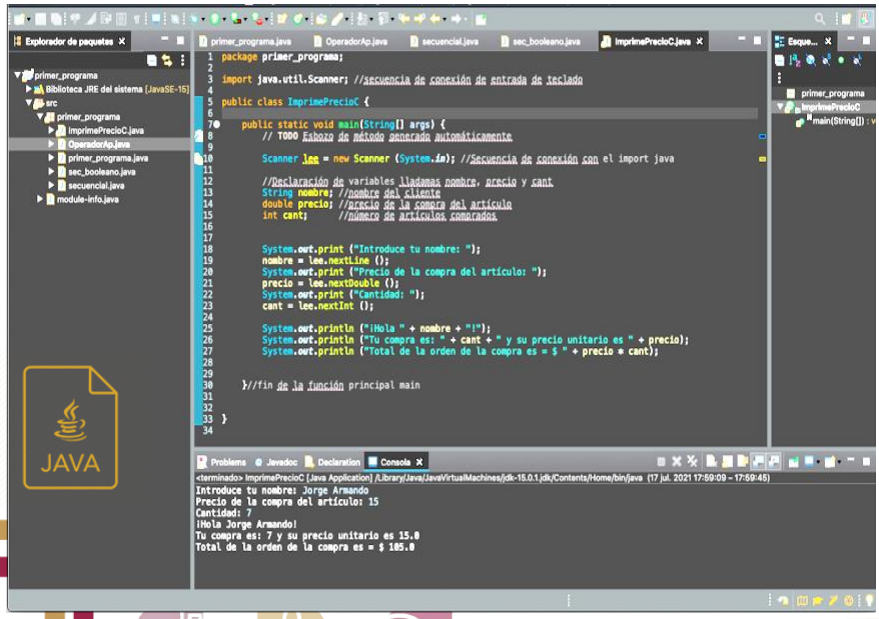
Existen al menos 3 manera para leer las entradas (datos por teclado) desde consola, ventanas o aplicaciones. Uno de ellos es usando el método **System.console().readLine();** mismo que debe asignarse a una variable tipo String. Ejemplo:

```
class EntradaTexto {

        public static void main(String[] args) {
            String nombre;
            System.out.print("Por favor, dime tu nombre: ");
            nombre = System.console().readLine();
            System.out.println("Hola " + nombre + ", ¡bienvenido a Java desde Cero!");
        }
}
```

Si en lugar de texto necesitamos datos numéricos, debemos convertir la cadena introducida en un número con el método adecuado. **Integer.parseInt()** convierte el texto introducido por teclado en un dato numérico, concretamente en un número entero.

La segunda opción es utilizando la Clase Buffered Reader, un método clásico de Java para leer datos de entrada, introducido en JDK 1.0. Este método se usa envolviendo System.in (flujo de entrada estándar) en un InputStreamReader que está envuelto en un BufferedReader, podemos leer la entrada del usuario en la línea de comando.

```
// Programa Java para demostrar BufferedReader
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
public class LecturaDatos
{
    public static void main(String[] args) throws IOException
    {
        //Ingrese datos usando BufferReader
        BufferedReader reader =
                  new BufferedReader(new InputStreamReader(System.in));

        // Leyendo datos usando readLine
        String name = reader.readLine();

        // Imprimir la línea de lectura
        System.out.println(name);
    }
}
```



El método más preferido para tomar datos de entrada. El objetivo principal de la clase Scanner es analizar los tipos primitivos y las cadenas con expresiones regulares, sin embargo, también se puede usar para leer las entradas del usuario en la línea de comandos.

Java proporciona una clase precompilada llamada *Scanner*, que permite obtener entradas ya sea del teclado o de un archivo. Cuando se hable de entrada o lectura de datos, se debe asumir que se está hablando de hacerlo a través del teclado.

La clase ***Scanner*** no es parte del conjunto fundamental de clases de Java (el core); por lo que, si el lector hace uso de ésta, necesita indicarle al compilador dónde encontrarla. Lo cual se realiza importando la clase en el programa. De manera más específica, se requiere incluir la sentencia **import** en la parte superior del programa (justo antes de la sección de inicio).

```
Import.java.util.Scanner;
```

Hay algo más que es necesario saber antes de preparar el programa para la lectura de datos. **Se requiere insertar esta sentencia en la parte superior del método main.**

```
Scanner stdIn = new Scanner (System.in);
```

La expresión *new Scanner (System.in)* crea un objeto. Como el lector ya sabe, un objeto almacena una colección de datos. En este caso, el objeto almacena caracteres introducidos por el usuario a través del teclado. La **variable** *stdIn* es una variable de referencia, y es inicializada a la dirección del objeto recientemente creado, *Scanner*. Después de la inicialización, la variable *stdIn* permite ejecutar operaciones de entrada de datos (lectura). Una vez establecido lo anterior, se puede leer y almacenar una línea de entrada llamando al método *nextLine* de la siguiente manera:

```
<variable> = stdIn.nestLine ();
```

Métodos de Entrada

La clase *Scanner* contiene muchos otros métodos que leen datos de entrada de diferentes formas. He aquí algunos de esos métodos:

nextInt( )	Se salta los espacios dejados en blanco hasta que encuentra un valor de tipo *int*. Devuelve un valor tipo *int*.
nextLong( )	Se salta los espacios dejados en blanco hasta que encuentra un valor de tipo *long*. Devuelve un valor tipo *long*.
nextFloat( )	Se salta los espacios dejados en blanco hasta que encuentra un valor de tipo *float*. Devuelve un valor tipo *float*.
nextDouble( )	Se salta los espacios dejados en blanco hasta que encuentra un valor de tipo *double*. Devuelve un valor tipo *double*.
next( )	Se salta los espacios dejados en blanco hasta que encuentra un token. Devuelve el token como un valor tipo *String*.

Lectura No. 5 "Instrucciones de Estructura"

Instrucciones: *Realiza la Lectura 5. "Instrucciones de Estructura" subrayando las ideas principales, y al finalizar realizar las prácticas de cada estructura de control*

LECTURA 5. "INSTRUCCIONES DE ESTRUCTURA"

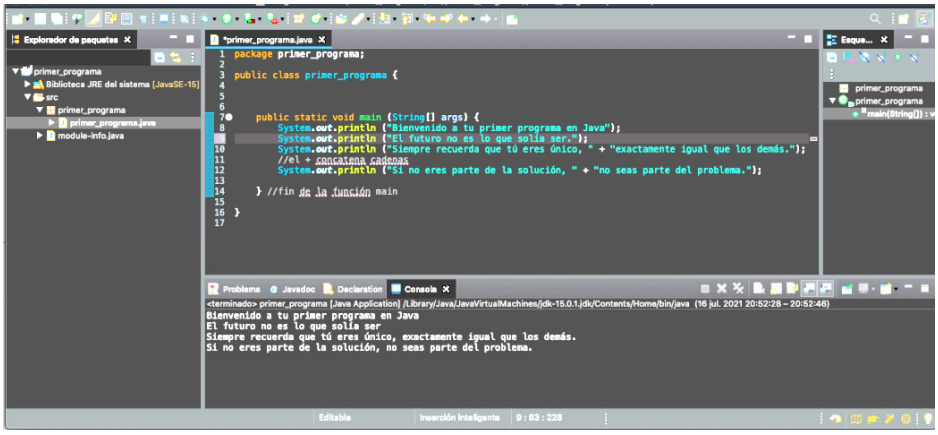
Es muy frecuente que, en la construcción de algoritmos y codificación, requieran del uso de diversos tipos de datos, que éstos tengan un comportamiento variable o constante, y que además deban ser manipulados de distintas maneras, o incluso tomar ciertas decisiones, por lo que es importante introducir algunos conceptos fundamentales que te permitan emplearlos adecuadamente. Por su naturaleza dentro de la computación en el tema de los algoritmos, códigos y programación, además de tener la naturaleza de C++; en Java permite al programador potencializar su creatividad y lógica. **Llamaremos estructura al bloque formado por una o más instrucciones que definen cierto comportamiento en el programa.** A continuación, estaremos estudiando las estructuras:

Estructura Secuencial
Estructura Condicional/Selectiva
Estructura Iterativa/Cíclica/Repetitiva

Antes de adentrarnos en cada estructura, es necesario considerar uno de los conceptos fundamentales en cualquier lenguaje de programación es el de variable. *Una variable es un espacio de memoria con un nombre asignado, al que el programa puede asignar un valor. El valor de la variable se puede cambiar durante la ejecución del programa.*

/* **primer_programa.java** Este programa imprime en pantalla un mensaje */

El método **println ( )** es como el método **print ( )**, excepto porque pone el carácter de línea nueva después de cada llamada. Para la salida de valores de cualquier tipo en Java se pueden utilizar ambos métodos, **print ( )** y **println ( )**.



«Estructura Secuencial»

Un algoritmo, un código en Java es un conjunto de pasos, instrucciones o acciones que se deben seguir, obedecer y ejecutar de manera ordenada para alcanzar un fin deseado. Las instrucciones o acciones se pueden realizar en forma secuencial, esto es, una después de otra.

En este bloque o estructura las acciones se ejecutan de manera sucesiva, una después de otra, siguiendo el orden físico en que aparecen las instrucciones. Aquí podemos encontrar instrucciones tales como Declaración o inicialización de variables, Operaciones de asignación, de Cálculo y Fin.

El siguiente programa muestra cómo se declara una variable y cómo se le asigna un valor. Además, también ilustra algunos aspectos nuevos de la salida por consola. Como indican los comentarios de las primeras líneas, el archivo correspondiente debe llamarse OperadorAp.java.

/* **OperadorAp.java** Este programa imprime los resultados de las operaciones básicas con 2 variables*/

```
package primer_programa;

public class OperadorAp {

    public static void main (String[] args) {
            System.out.println ("Operaciones con enteros");
            int ia=, ib=;
            int isuma, iresto;
            isuma=ia+ib;
            System.out.println ("El resultado de la suma es: " +isuma);

            int iproducto = ia*ib;
            System.out.println ("El resultado del producto es: " +iproducto);
            System.out.println ("El resultado del cociente es: " + (ia/ib));
            iresto=ia%ib;
            System.out.println ("El resto de la división entera es " +iresto);
            System.out.println("*****************************************");

            System.out.println("Operaciones con números decimales");
            double da=7.5, db=3.0;

            double dsuma=da+db;
            System.out.println("El resultado de la suma es "+dsuma);
            double dproducto=da*db;
            System.out.println("El resultado del producto es "+dproducto);
            double dcociente=da/db;
            System.out.println("El resultado del cociente es "+dcociente);
            double dresto=da%db;
            System.out.println("El resto de la división es "+dresto);
            //el + concatena cadenas

        } //fin de la función main
}
```

```
Operaciones con enteros
El resultado de la suma es: 10
El resultado del producto es: 21
El resultado del cociente es: 2
El resto de la división entera es 1
*****************************************
Operaciones con números decimales
El resultado de la suma es 10.5
El resultado del producto es 22.5
El resultado del cociente es 2.5
El resto de la división es 1.5
```

Al ejecutar este programa, se obtiene la siguiente salida:

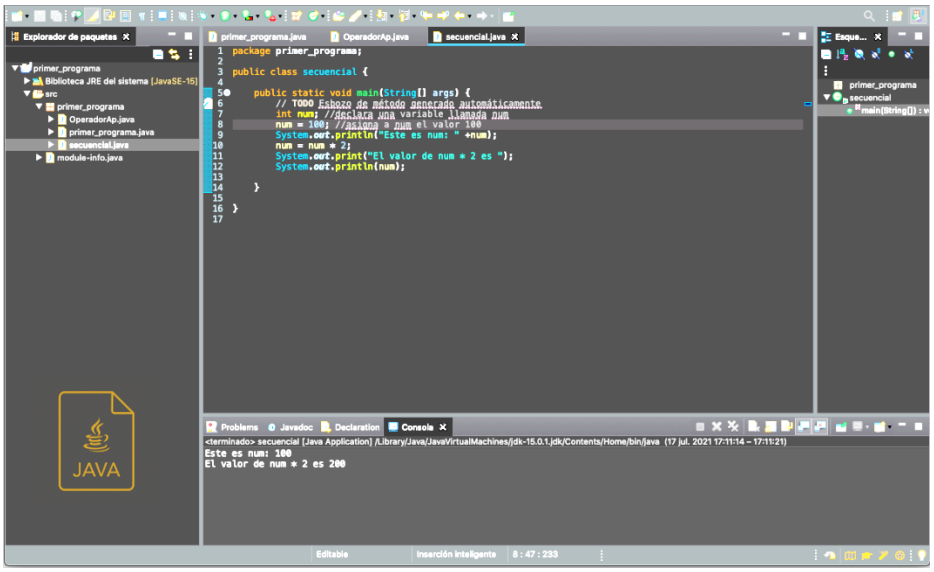
Se declaran dos variables de tipo **entero (int)** y se les asigna un valor diferente a cada una: *ia=7, ib=3;* para hacer las operaciones aritméticas con éstas.

El método *System.out.println( );* permite imprimir en pantalla un mensaje para posteriormente hacer un salto de línea (↵). En este ejemplo lo que se encuentra después de las comillas dobles, expresa la concatenación de la oración entre comillas con el valor que se encuentra en la variable que sucede al signo (+). Ejemplo:

System.out.println("El resultado del producto es: " +iproducto);

Nota: Las declaraciones de las variables puedes hacerlas en una misma línea si corresponden al mismo tipo de dato.

/* **secuencial.java** Este programa imprime en pantalla el doble de un número */



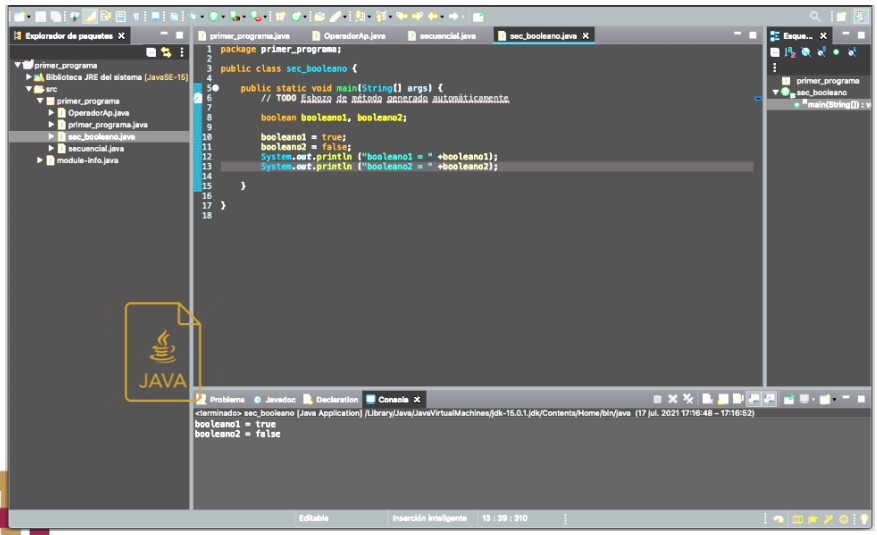
En el programa, la línea num = 100; // asigna asigna a num el valor 100.

En Java, el operador de asignación es el signo igual. La siguiente línea del código es la responsable de desplegar el valor de num precedido por la cadena de caracteres "Esto es num:".

System.out.println("Este es num: " +inum);

En esta sentencia, el signo de suma hace que el valor de num sea añadido a la cadena que le precede, y a continuación se despliega la cadena resultante. Lo que realmente ocurre es que num se convierte en el carácter equivalente y después se concatena con la cadena que le precede. Utilizando el operador +, se pueden encadenar tantos elementos como se desee dentro de una única sentencia println ( ).

La siguiente línea de código asigna a num el valor de num multiplicado por dos. Como en otros lenguajes, Java utiliza el operador * para indicar multiplicación. Después de la ejecución de esta línea, el valor almacenado en num será 200.



La línea
public static void main (String [ ] args)
corresponde a la función principal pública, estática en la que se codifican las instrucciones que llamaran a otras funciones o métodos.

Aquí se usan dos variables de timpo **boolean (verdadero o falso)** y lo único que hace el programa es mostrar los valores asignados a las variables.

/* **sec_booleano.java** Este programa imprime el valor de dos variables booleanas */

«Estructura Condicional»

En algoritmos un poco más complejos es usual tomar alguna decisión. Del resultado de esa decisión se establece qué proceso realizar, o un camino alternativo a seguir. La estructura selectiva, de selección o condicional básicamente puede ser simple o doble y anidada.

►LA SECUENCIA IF

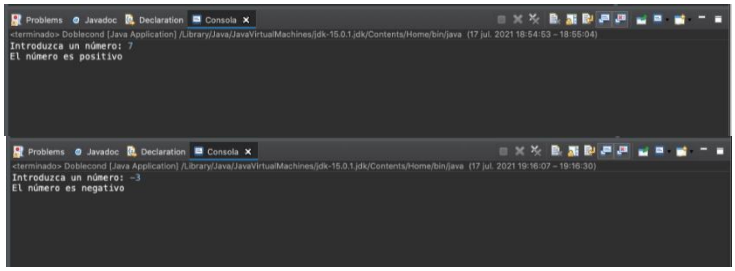
La sentencia **if** de Java actúa de la misma forma que la sentencia IF en cualquier otro lenguaje. Además, es sintácticamente idéntica a las sentencias **if** de C, C++ y C#. A continuación, se presenta su forma más simple.

If (condición) sentencia;

donde *condición* es una expresión booleana. Si la *condición* es verdadera, entonces se ejecuta la sentencia. Si la *condición* es falsa, entonces se evita la sentencia. A continuación se presenta un ejemplo:

/* **Doblecond.java** Este programa lee un número e imprime si es positivo o negativo */

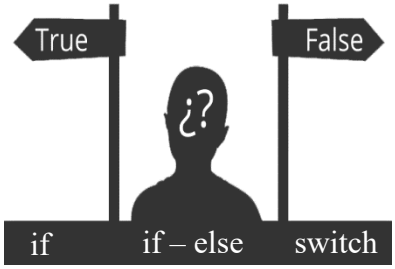
```
package primer_programa;

import java.util.Scanner;

public class Doblecond {


    public static void main(String[] args) {
        // TODO Esbozo de método generado automáticamente
        Scanner lector = new Scanner (System.in);
        int num;
        System.out.print ("Introduzca un número: ");
        num = lector.nextInt();
        if (num < 0) {
            System.out.println("El número es negativo");
        }
        else {
            System.out.println("El número es positivo");
        }


    }
}
```



Aquí analizamos un número leído para compartir en pantalla si es negativo o positivo, donde la línea de la condición *if(num < 0)* se expresa una pregunta como *"¿el número es menor que cero?"* o *"¿el número es negativo?"* por lo que el código puede mostrar una de dos salidas por la sentencia *else* como decir *"sino"*.

NOTA IMPORTANTE

Cuando una sentencia *if* incluye dos o más sentencias subordinadas, se deben encerrar entre paréntesis de llave. Dicho de otra manera, se debe utilizar un *bloque*. Un bloque, también llamado *sentencia compuesta*, es un conjunto de cero o más sentencias rodeadas por llaves. Se puede utilizar un bloque en cualquier parte en que una sentencia estándar pueda ser utilizada. Si no se emplean paréntesis de llave para dos sentencias subordinadas al *if*, la computadora considera sólo la primera como subordinada al *if*. Cuando se supone que es sólo una sentencia subordinada, no es necesario encerrarla entre llaves, pero se recomienda hacerlo de cualquier manera. Así, no se tendrán problemas cuando haya que volver al código e insertar sentencias subordinadas adicionales en ese punto del programa.



►TRES FORMAS DE LA SENTENCIA if

Existen tres formas básicas de una sentencia if:

- "if", se utiliza cuando se quiere hacer una cosa o no hacer nada.
- "if, else", se utiliza cuando se quiere hacer una u otra cosa.
- "if, else if", se utiliza cuando hay tres o más posibilidades.

Y a continuación, la forma en que el "if, else" funciona:

- Si la condición es verdadera, ejecutar las sentencias subordinadas al "if", y saltarse las sentencias subordinadas al "else".
- Si la condición es falsa, saltarse la sentencia(s) subordinada(s) al "if", y ejecutar todas las sentencias subordinadas al "else".

A continuación, se ofrece un ejemplo que utiliza la sentencia con el formato "if, else":

```
package primer_programa;

import java.util.Scanner;

public class Divisible {

    public static void main(String[] args) {
        // TODO Esbozo de método generado automáticamente

        int n, d;

        Scanner entrada = new Scanner (System.in);
        System.out.println("Introduzca el primer valor: ");
        n = entrada.nextInt();
        System.out.println("Introduzca el segundo valor: ");
        d = entrada.nextInt();
        if (n % d == 0) {
            System.out.println(n + " es divisible entre " +d);
        }
        else {
            System.out.println(n + "no es divisible entre " +d);
        }


    } //fin de la función principal main
} //fin de la clase Divisible
```

```
Problems  Javadoc  Declaration  Consola
<terminado> Divisible [Java Application] /Library/Java/JavaVirtualMachines/jdk-15.0.1.jdk/Contents/Home/bin/java  (17 jul. 2021 19:31:37 – 19:32:09)
Introduzca el primer valor:
12
Introduzca el segundo valor:
3
12 es divisible entre 3
```

```
Problems  Javadoc  Declaration  Consola
<terminado> Divisible [Java Application] /Library/Java/JavaVirtualMachines/jdk-15.0.1.jdk/Contents/Home/bin/java  (17 jul. 2021 19:33:06 – 19:33:47)
Introduzca el primer valor:
7
Introduzca el segundo valor:
3
7no es divisible entre 3
```

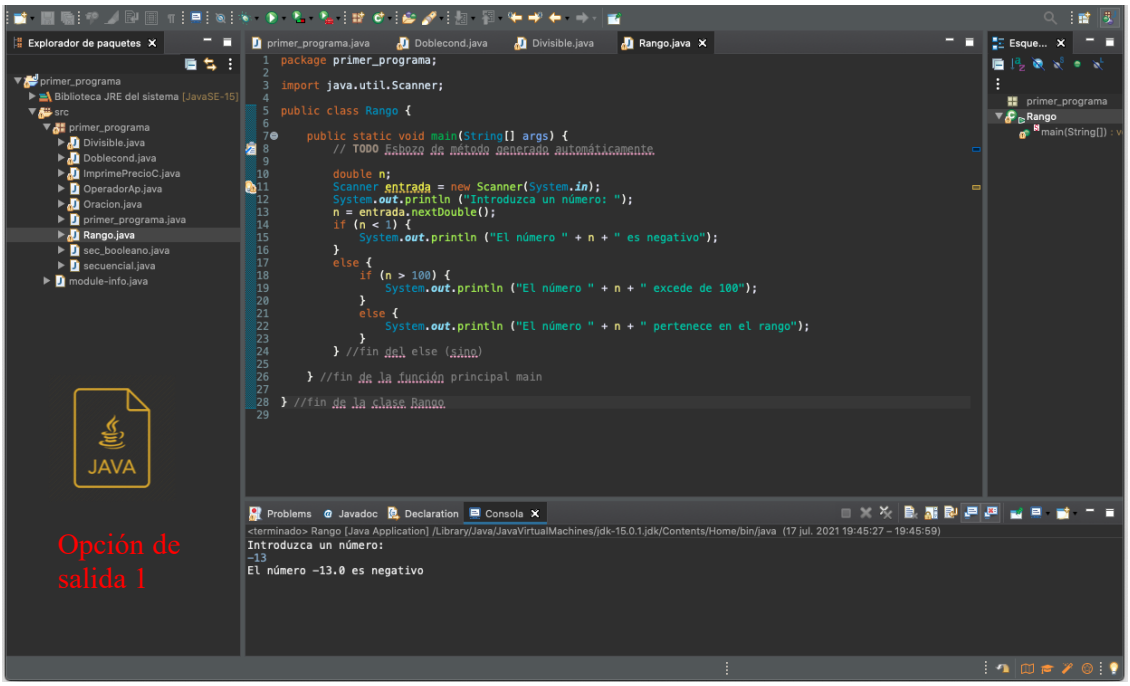
En este programa, se leen dos números para determinar si el primer número es divisible por el segundo número leído.

En la condición se usa el operador ***modulardor***

En casos en los que se requiere analizar más de una decisión, es posible aplicar las **condiciones anidadas** en las que se puede observar después de la primera expresión/condición con su respectivo *else* para ser seguido de un nuevo *if* con su *else*, y éste puede continuar con la misma estructura las veces que requiera.

Una sentencia *if* es anidada cuando alguna de las ramas, sin importar si es verdadera o falsa, también es *if*; entonces se puede utilizar para tomar decisiones con varias opciones o multi opciones.

/* **Rango.java** Este programa lee un número y determina si pertenece al rango de 1 al 100 */



En este programa, se solicita un número de tipo double al usuario, debido a que el número puede ser corto o largo, decimal o entero, positivo o negativo; después de la lectura, recuerda que debe ser almacenado en la variable para luego comparar:

Condición 1.- If (n < 1)

Condición 2.- If (n > 100)

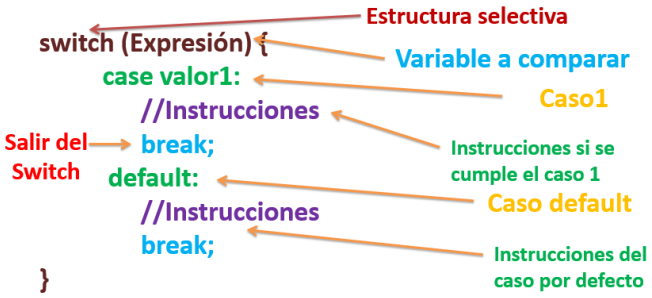
El rango considerado es [1, 100], es por eso que la condición 1 expresa que ***si*** el número leído es menor que cero, el número es negativo y fuera del rango. La condición 2 expresa que ***si*** el número leído es mayor que cien, el número está fuera del rango, ***sino*** el número pertenece al rango.

►LA SECUENCIA DE CONTROL SWITCH

En Java, ***switch*** es una sentencia que se utiliza para elegir una de entre múltiples opciones; es especialmente útil cuando la selección se basa en el valor de una variable simple o de una expresión simple denominada ***expresión de control o selector***; el valor de dicha expresión puede ser de tipo *int* o *char* pero no de tipo *double*.

SINTAXIS

```
switch (selector)
{
   case etiqueta1 : sentencias1 ;
     break;
   case etiqueta2 : sentencias2 ;
     break;
   .
   .
   .
   case etiqueta_n : sentencias_n ;
     break;
   default: sentenciasd ; // opcional
}
```



La expresión de control o selector se evalúa y compara con cada una de las etiquetas de *case*; además, debe ser un tipo ordinal, por ejemplo, *int*, *char*, *bool* pero no *float* o *string*; cada etiqueta es un valor único, constante y debe tener un valor diferente de los otros. Si el valor de la expresión es igual a una de las etiquetas *case*, por ejemplo, *etiqueta1*, entonces la ejecución comenzará con la primera sentencia de *sentencias1* y continuará hasta encontrar *break* o el final de *switch*.

El tipo de cada etiqueta debe ser el mismo que la expresión de selector; las expresiones están permitidas como etiquetas, pero sólo si cada operando de la expresión es por sí misma una constante; por ejemplo, 4, +8 o m*15, siempre que **m** hubiera sido definido anteriormente como constante nombrada.

Si el valor del selector no está listado en ninguna etiqueta case no se ejecutará ninguna de las opciones a menos que se especifique una acción predeterminada. La omisión de una etiqueta default puede crear un error lógico difícil de prever; aunque ésta es opcional, se recomienda su uso, a menos de estar absolutamente seguro de que todos los valores de selector están incluidos en las etiquetas case.

Comparación de las sentencias *if-else-if* y *switch*; se quiere determinar si un carácter

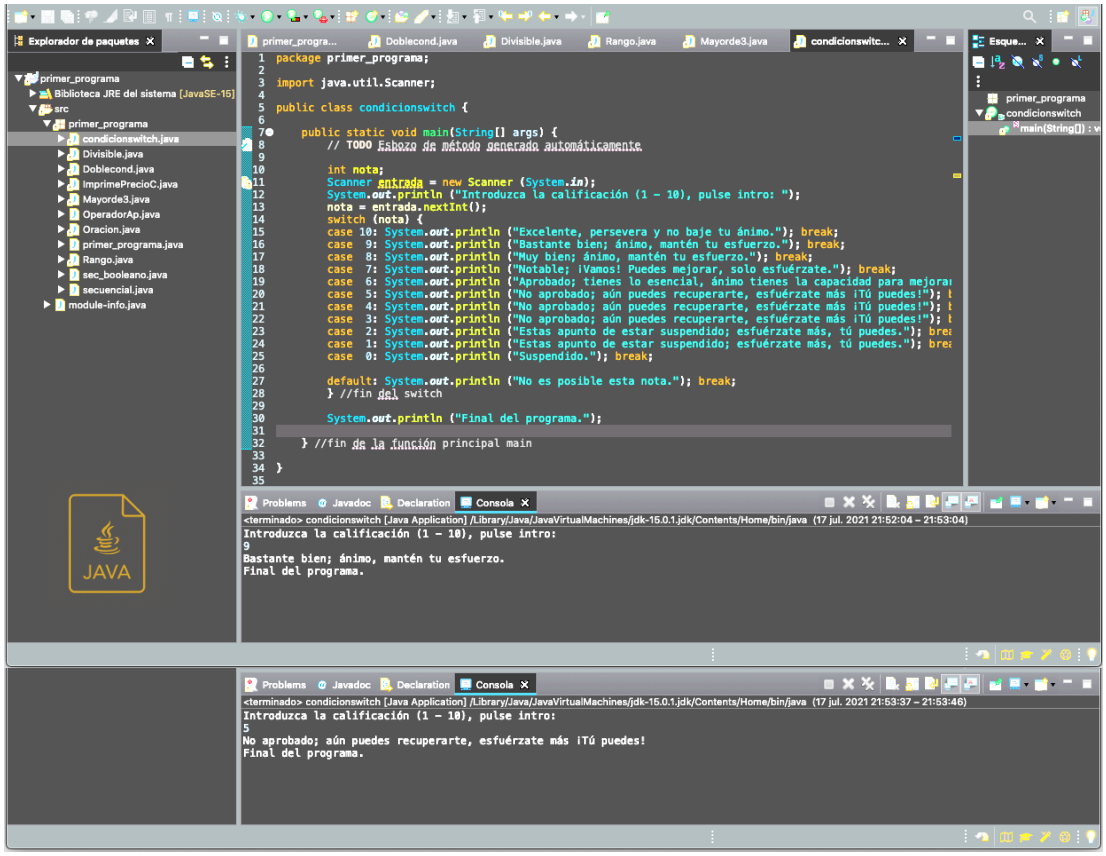
Solución con *if-else-if*.

```
if ((car == 'a') || (car == 'A'))
  System.out.println(car + " es una vocal");
else if ((car == 'e') || (car == 'E'))
        System.out.println(car + " es una vocal");
    else if ((car == 'i') || (car == 'I'))
          System.out.println(car + " es una vocal");
        else if ((car == 'o') || (car == 'O'))
                System.out.println(car + " es una vocal");
            else if ((car == 'u') || (car == 'U'))
                    System.out.println(car + " es una vocal");
                  else
                    System.out.println(car + " no es una vocal");
```

Solución con *switch*.

```
switch (car) {
case 'a': case 'A':
case 'e': case 'E':
case 'i': case 'I':
case 'o': case 'O':
case 'u': case 'U':
System.out.println(car + " es una vocal");
break;
default:
System.out.println(car + " no es una vocal"); }
```

/* **condicionswitch.java** En este programa se considera un rango entre 1 y 10 para asignar la nota de un curso, el programa ilustra la selección múltiple con la sentencia *switch*. */



«Estructura Cíclica o Repetitiva»

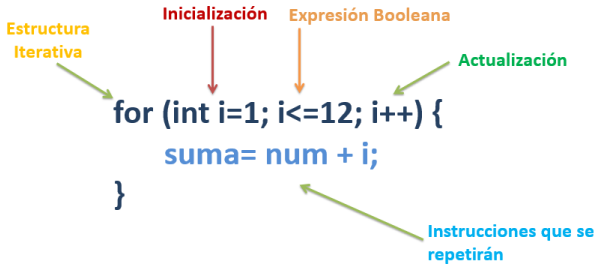
Las estructuras de control de bucles, repiten la ejecución de una secuencia particular de sentencias. Si se requiere ejecutar un bloque de código muchas veces, por supuesto que se podría escribir de manera repetitiva el código como se necesitara. Sin embargo, esto conduce a la redundancia, lo cual es algo que se quiere evitar en un programa de cómputo, porque abre la puerta a la inconsistencia. Es mejor escribir el código una vez y reutilizarlo. La forma más simple de reutilizar un bloque de código es ir hacia atrás en donde inicia dicho bloque, y ejecutarlo una vez más. Eso se denomina un bucle o ciclo. Cada ciclo tiene una condición que determina cuántas veces se repetirá el mismo.

Un bucle o lazo es cualquier construcción de programa que repite una sentencia o secuencia de sentencias determinado número de veces; cuando ésta se menciona varias veces en un bloque se denomina cuerpo del bucle; cada vez que éste se repite se denomina iteración del bucle. Las dos cuestiones principales de diseño en la construcción del bucle son: ¿cuál es el cuerpo del bucle? y ¿cuántas veces se iterará el cuerpo del bucle?

Una característica que aumenta considerablemente la potencia de las computadoras es su capacidad para resolución de algoritmos repetitivos con gran velocidad, precisión y fiabilidad, mientras que para las personas las tareas repetitivas son difíciles y tediosas de realizar; esta sección cubre las estructuras de control iterativas o repetitivas, cíclicas de acciones; Java soporta tres tipos de ellas: los bucles *for*, *while* y *do-while*; todas éstas controlan el número de veces que una sentencia o listas de sentencias se ejecutan.

►BUCLE FOR

Un ciclo *for* es apropiado cuando se sabe el número exacto de iteraciones antes de que inicie. Éste se divide en tres componentes: *inicialización*, *condición* y *actualización de componentes*. La siguiente lista explica cómo el ciclo for utiliza esos tres componentes.

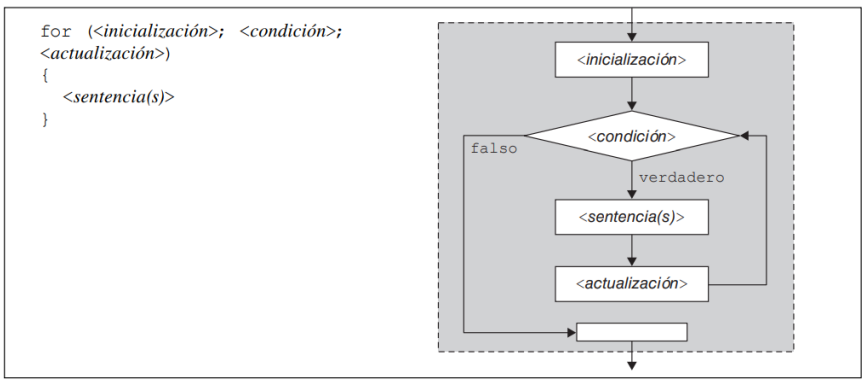


1. **Inicialización de componente:** Antes de que se ejecute la primera instrucción dentro del ciclo, ejecutar la inicialización del componente.
2. **Condición del componente:** Antes de que se realice cualquier iteración, evaluar la condición del componente:
   - Si la condición es *verdadera*, ejecutar el cuerpo del ciclo.
   - Si la condición es *falsa*, terminar el ciclo (pasar a la sentencia debajo del paréntesis de llave de cierre del ciclo).
3. **Actualización del componente:** Después de cada ejecución dentro del cuerpo del ciclo, volver al encabezado del ciclo y ejecutar la actualización del componente. Posteriormente, volver a verificar la continuación de la condición del segundo componente, y si se satisface, ir al cuerpo del ciclo otra vez.

La sintaxis de la sentencia *for* es:

```
for (inicialización; condición de terminación; incremento)
    Sentencias;
```

Las *sentencias* podrán ser cero, una única sentencia o un bloque, y serán lo que se repita durante el proceso del bucle.

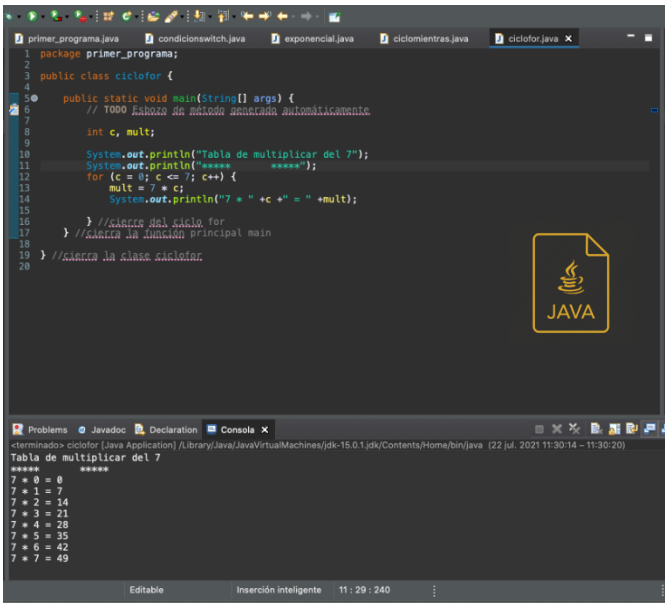


La *inicialización* fija los valores iniciales de la variable o variables de control antes de que el bucle *for* se procese y ejecute solo una vez. Si se desea inicializar más de un valor, se puede utilizar un operador especial de los bucles *for* en Java, el operador coma, para pegar sentencias. Cuando no se tiene que inicializar, se omite este apartado; sin embargo, nunca se debe omitir el punto y coma que actúa como separador.

La *condición de terminación* se comprueba antes de cada iteración del bucle y éste se repite mientras que dicha condición se evalúe a un valor verdadero. Si se omite no se realiza ninguna prueba y se ejecuta siempre la sentencia *for*.

El *incremento* se ejecuta después de que se ejecuten las *sentencias* y antes de que se realice la siguiente prueba de la *condición de terminación*. Normalmente esta parte se utiliza para incrementar o decrementar el valor de la/las variables de control y, al igual que en la inicialización, se puede usar en ella el operador coma para pegar sentencias. Cuando no se tienen valores a incrementar se puede suprimir este apartado.

En esencia, el bucle *for* comprueba si la *condición de terminación* es verdadera. Si la condición es *Verdadera*, se ejecutan las sentencias del interior del bucle, y si la condición es falsa, se saltan todas las sentencias del interior del bucle, es decir, no se ejecutan. Cuando la condición es *verdadera*, el bucle ejecuta una iteración (todas sus sentencias) y a continuación la variable de control del bucle se incrementa.

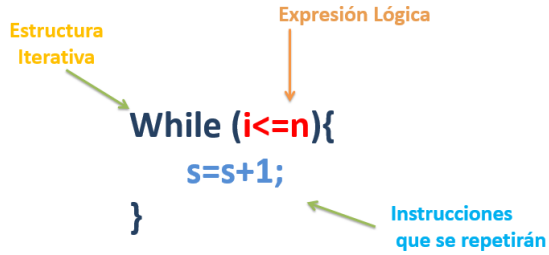


/* **ciclofor.java** Este programa presenta la tabla de multiplicar del 7, comenzando del 0 hasta 7.
*/

►BUCLE WHILE

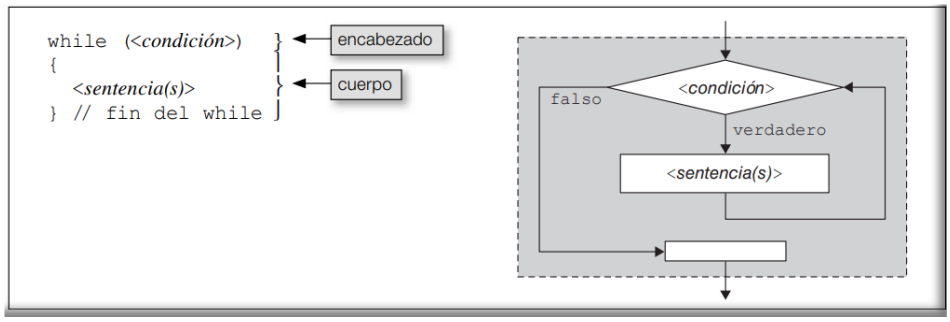
Un bucle *while* tiene una condición, una expresión lógica que controla la secuencia de repetición; su posición es delante del cuerpo del bucle y significa que *while* es un bucle *pretest*, de modo que cuando éste se ejecuta, se evalúa la condición antes de ejecutarse el cuerpo del bucle.

La ejecución de la *sentencia* o *sentencias* expresadas se repite mientras la condición del bucle permanece *verdadera* y termina al volverse *falsa*; también indica que la condición se examina antes de ejecutarse el cuerpo y, por consiguiente, si aquella es inicialmente falsa, éste no se ejecutará; en otras palabras, el cuerpo de un bucle *while* se ejecutará cero o más veces.



La variable que representa la condición del bucle también se denomina *variable de control* debido a que su valor determina si el cuerpo se repite; ésta debe ser: 1) inicializada, 2) comprobada y 3) actualizada para que aquél se ejecute adecuadamente; cada etapa se resume así:

1. *Inicialización*. Se establece contador a un valor inicial antes de que se alcance la sentencia while, aunque podría ser cualquiera, generalmente es 0.
2. *Prueba/condición*. Se comprueba el valor de contador antes de la iteración; es decir, el comienzo de la repetición de cada bucle.
3. *Actualización*. Durante cada iteración, contador actualiza su valor incrementándose en uno mediante el operador ++.



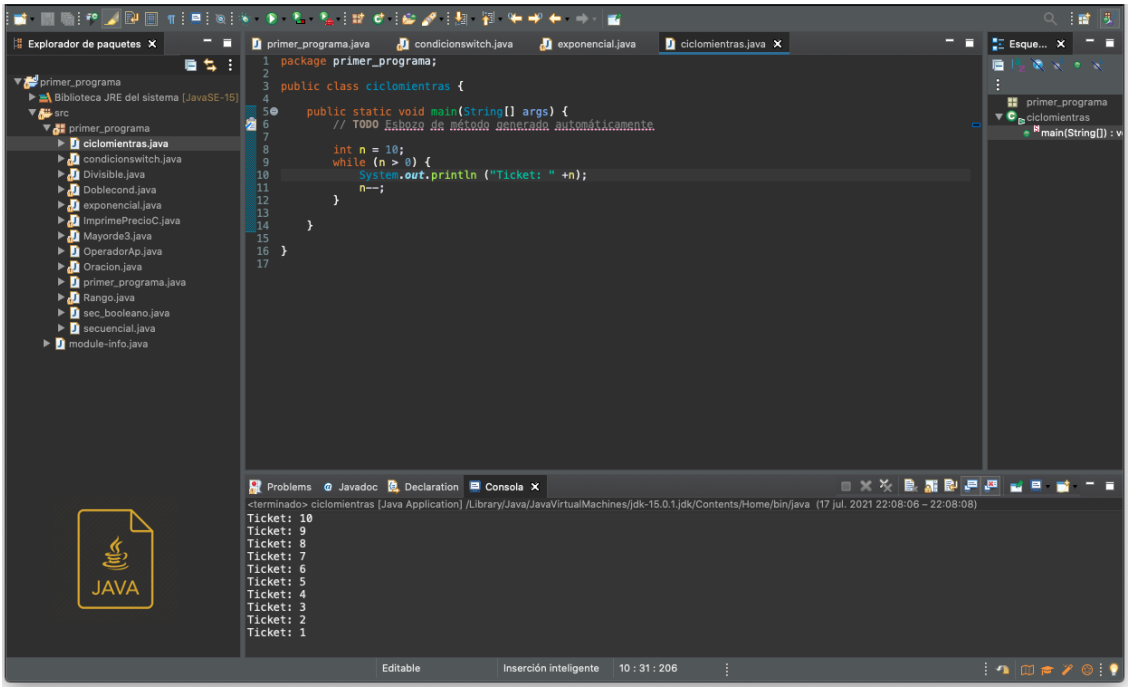
Si la variable de control no se actualiza se ejecutará un bucle infinito; en otras palabras, esto sucede cuando su condición permanece sin hacerse falsa en alguna iteración.

Por ejemplo:

```
// bucle infinito
contador = 1; while (contador < 100)
{ System.out.println (contador);
  contador-- ; à decrementa en 1 contador
}
```

Se inicializa contador a 1 (menor que 100), como contador-- decrementa en 1 el valor de contador en cada iteración su valor nunca llegará a 100, número necesario para que la condición sea falsa; por consiguiente, contador < 100 siempre será verdadera, resultando un bucle infinito.

/* **ciclomientras.java** Ejemplo del funcionamiento del ciclo while. */



En este ejemplo se observa en la condición **n > 0** (n es mayor que cero), expresa que el número **n** debe ser positivo, mayor que 0. El contador es la misma variable **n**, donde se *decrementa* mientras el ciclo se cumple. Al final se imprime en pantalla la oración "Ticket: 10" … "Ticket: 1".

►BUCLE DO WHILE

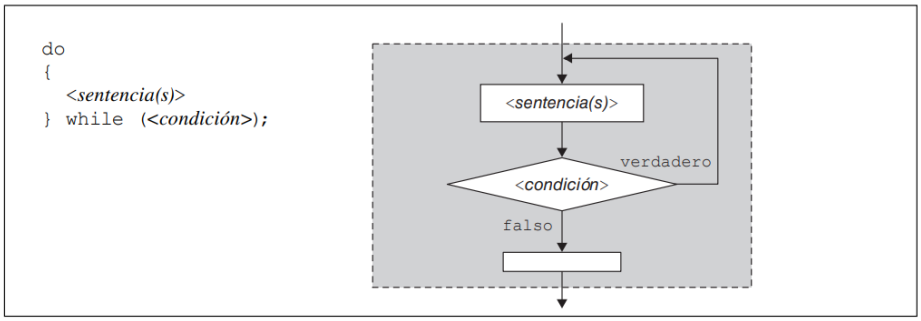
Como se acaba de ver, si la expresión condicional que controla un ciclo **while** es inicialmente falsa, el cuerpo del ciclo no se ejecutará ni una sola vez. Sin embargo, puede haber casos en los que se quiera ejecutar el cuerpo del ciclo al menos una vez, incluso cuando la expresión condicional sea inicialmente falsa. En otras palabras, puede que se desee evaluar la expresión condicional al final del ciclo, en lugar de hacerlo al principio. Afortunadamente, Java dispone de un ciclo que lo hace exactamente así, el ciclo **do-while**. El ciclo **do-while** ejecuta siempre, al menos una vez, el cuerpo, ya que la expresión condicional se encuentra al final. Su forma general es:

```
do {
   // cuerpo del ciclo
} while (condición);
```

En cada iteración del ciclo **do-while** se ejecuta en primer lugar el cuerpo del ciclo, y a continuación se evalúa la expresión condicional. Si la expresión es verdadera, el ciclo se repetirá. En caso contrario, el ciclo finalizará. Como en todos los demás ciclos de Java, la condición debe ser una expresión **booleana**.

He aquí cómo trabaja el ciclo do:

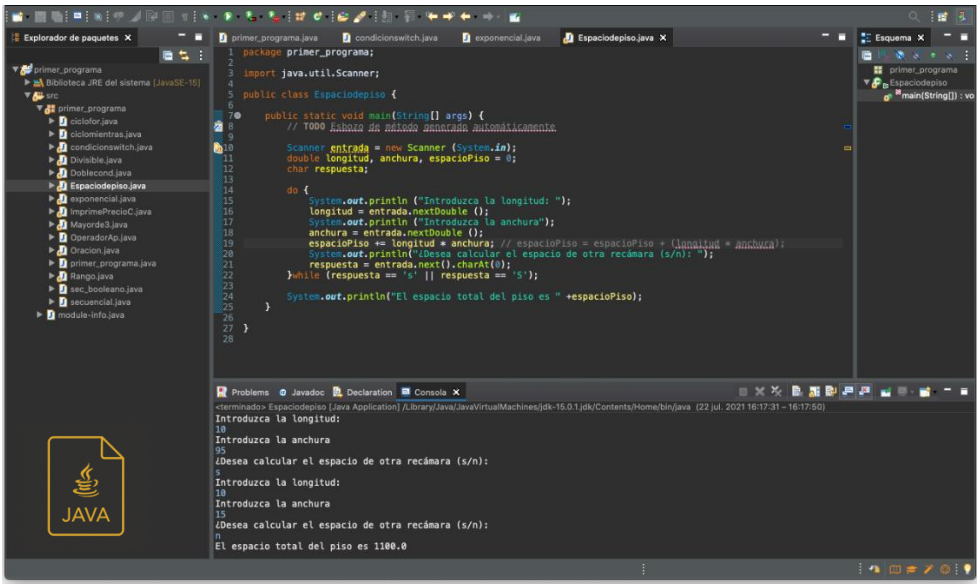
1. Ejecuta el cuerpo del ciclo do.
2. Verifica la condición final.
3. Si la condición es verdadera, vuelve a la parte de arriba del ciclo do y repite el paso 1.
4. Si la condición es falsa continúa con la sentencia que aparece inmediatamente después del ciclo.



Ahora se ilustrará el ciclo do mediante un problema de ejemplo. Suponga que se le pide escribir un programa que solicite al usuario introducir las dimensiones de longitud y la anchura de cada recámara de una casa, con el fin de calcular el espacio total de la casa completa. Después de introducir cada longitud/anchura, se le pregunta al usuario si existe otra recámara. Cuando no hay más recámaras, se imprime la dimensión del espacio de piso total.

Para resolver este problema, lo primero que hay que preguntarse es si realmente se requiere un ciclo. ¿Se requiere repetir algo? Sí, se deseará leer las dimensiones de manera repetitiva, para que el ciclo sea apropiado. Para determinar el tipo de ciclo hay que preguntarse: ¿se requerirá la lectura de las dimensiones de la recámara al menos una vez? Sí, cada casa debe tener al menos una, por lo que se necesitará leer al menos un conjunto de dimensiones. Entonces, es apropiada la utilización de un ciclo **do while** para este problema.

/* **Espaciodepiso.java** Este programa calcula el espacio total de piso en una casa. */



Observe que la parte de la condición *while* está en la misma línea que la llave de cerrado: esto es un buen estilo. Es posible poner un *while (<condición>);* en la línea después del cierre de llaves, pero esto es un mal estilo porque parecería que se está intentando iniciar un ciclo *while*.

BIBLIOGRAFÍA DE LA LECTURA 6

Programación en Java 6, algoritmos y programación orientada a objetos. Luis Joyanes Aguilar, Ignacio Zahonero Martínez, McGrawn-Hill, 2011, México D.F., pág. 20.

Jorge Martínez Ladron de Guevara, G- Tec. (2020). Fundamentos de programación en Java (Spanish Edition). Madrid, España: Independently published.

Java 2 Manual de Programación, Luis Joyanes Aguilar, Matilde Fernández, Osborne McGrawn-Hill, Salamanca, España, pág. XIV.

Aprendiendo Java y Programación Orientada a Objetos. Gustavo Guillermo Pérez. 2008. Obtenido de: https://www.clubensayos.com/Ciencia/Aprendiendo-Java-y-Programaci%C3%B3n-Orientada-a-Objetos/262071.html

Manual de Referencia Java, Herbert Schildt, McGrawHill, 2009, Ciudad de México, pág. 9.

https://www.programarya.com/

Estructura SIMPLE-SECUENCIAL
Práctica No. 01 "Promedio de 3 Calificaciones"

PROPÓSITO: Utilizando la estructura simple y el código de la práctica 1 de C++, convirtiendo el código de C++ a Java, distinguiendo las instrucciones que son necesarias para la realización de programas.

CONOCIMIENTOS

- ✓ Estructura Simple
- ✓ Uso del IDE Eclipse
- ✓ Clase Scanner
- ✓ Función principal
- ✓ Compilar y ejecutar
- ✓ Guardar código

DESARROLLO

1. Realiza un programa de estructura secuencial, que solicite escribir tres calificaciones y a partir de ello realice el cálculo del promedio de ellas.
2. Abre el editor del Lenguaje Java, Eclipse.
3. Guarda el proyecto o la clase en tu carpeta con la siguiente estructura: Practica02, siglas de tu nombre, grupo y turno. Ejemplo: **"Practica02_JAFPGV"** donde JAFP son las siglas de tu nombre, G es el grupo y V el turno para Matutino y V vespertino.
4. Escribe el código en el editor con apoyo de tu profesor.
5. Posteriormente compilamos el programa, presiona el botón de compilar  o utilizar la combinación de teclas **ctrl + espacio**.

6. A continuación, ejecutamos el programa, presiona el botón de ejecutar  o utilizar la combinación de teclas **ctrl + F11** se genera el archivo ejecutable de nuestro programa.

```java
package primer_programa;

import java.util.Scanner;

public class Sec_Promedio {

    public static void main(String[] args) {
        // TODO Esbozo de método generado automáticamente

        Double Cal1, Cal2, Cal3, Promedio;

        Scanner lee = new Scanner (System.in);

        System.out.println("Ingresa la primer calificación: ");
        Cal1 = lee.nextDouble();

        System.out.println("Ingresa la segunda calificación: ");
        Cal2 = lee.nextDouble();

        System.out.println("Ingresa la tercera calificación: ");
        Cal3 = lee.nextDouble();

        Promedio = (Cal1 + Cal2 + Cal3) / 3;

        System.out.println("El promedio es: " +Promedio);

    }
}
```

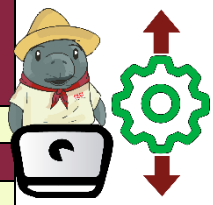

```
Problems  Javadoc  Declaration  Consola
<terminado> Sec_Promedio [Java Application] /Library/Java/JavaVirtualMachines/jdk-15.0.1.jdk/Contents/Home/bin/java  (24 jul. 2021 17:52:58 – 17:53:24)
Ingresa la primer calificación:
9
Ingresa la segunda calificación:
10
Ingresa la tercera calificación:
9
El promedio es: 9.333333333333334
```

7. **CONCLUSIONES:** Al finalizar la práctica, con tus palabras contesta las siguientes preguntas:

¿Qué aprendí?

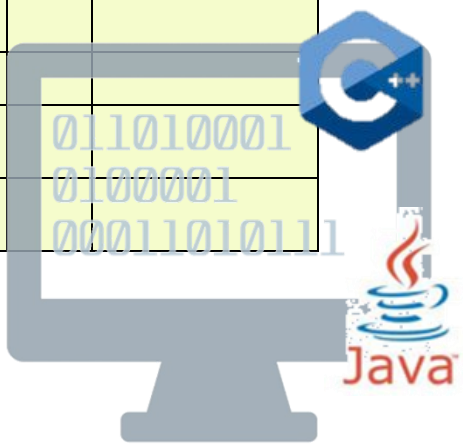
¿Dónde lo aplico en mi vida cotidiana y académica?

INSTRUMENTO DE EVALUACIÓN
LISTA DE COTEJO
Práctica 01. "Promedio de 3 Calificaciones"

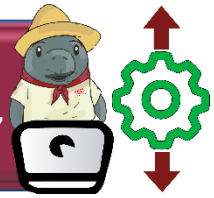


DATOS GENERALES	
Nombre(s) del alumno(s)	**Matricula(s)**
Producto: Programa de Promedio de 3 Calificaciones en Java.	**Fecha**
Submódulo II: Programación en Java	**Periodo: 2026 – 2027A**
Nombre del docente	**Firma del docente**

CRITERIO	INDICADORES	VALOR OBTENIDO		PONDERACIÓN	CALIF	OBSERVACIONES Y/O SUGERENCIAS DE MEJORA
		SI	NO			
1	Codifica correctamente la secuencia lógica del desarrollo del programa.			**2**		
2	Identifica y aplica correctamente los métodos e instrucciones.			**2**		
3	Guarda el archivo con la nomenclatura solicitada.			**1**		
4	Compila y ejecuta el programa, mostrando en pantalla el resultado del cálculo del promedio de tres calificaciones.			**3**		
5	Entrega en tiempo y forma.			**1**		
6	Presenta conclusiones de la práctica.			**1**		
				CALIFICACIÓN		



Estructura CONDICIONAL O DECISIÓN
Práctica No. 02"Determinar de Tres Números Dados Cuál es el Mayor, Cuál el Menor y Realizar el Producto de Éstos"



PROPÓSITO: Utiliza la estructura condicional y analiza sus ventajas y desventajas, teniendo en cuenta su utilidad en su vida escolar y social.

CONOCIMIENTOS

- ✓ Estructura IF ELSE
- ✓ Clase Scanner
- ✓ Variable inicial
- ✓ Variable final
- ✓ Condición

DESARROLLO

1. Realiza utilizando la estructura condicional, un programa que solicite el ingreso de tres números cualquiera, determine cuál de ellos es el mayor, cual el menor y calcular el producto de los tres números para presentar el resultado en pantalla.
2. Abre el editor del Lenguaje Java, Eclipse.
3. Guarda el proyecto o la clase en tu carpeta con la siguiente estructura: Practica03, siglas de tu nombre, grupo y turno. Ejemplo: "**Practica03_JAFPGV**" donde JAFP son las siglas de tu nombre, G es el grupo y V el turno para Matutino y V vespertino.
4. Escribe el código en el editor con apoyo de tu profesor.
5. Posteriormente compilamos el programa, presiona el botón de compilar  o utilizar la combinación de teclas **ctrl + espacio**.
6. A continuación, ejecutamos el programa, presiona el botón de ejecutar  o utilizar la combinación de teclas **ctrl + F11** se genera el archivo ejecutable de nuestro programa.

```java
package primer_programa;

import java.util.Scanner;

public class Practica02_JAFPGV {

    public static void main(String[] args) {
        // TODO Esbozo de método generado automáticamente

        Double a, b, c, producto;

        Scanner lee = new Scanner (System.in);


        System.out.println("Escribe el primer número: ");
        a = lee.nextDouble();

        System.out.println("Escribe el segundo número: ");
        b = lee.nextDouble();

        System.out.println("Escribe el tercer número: ");
        c = lee.nextDouble();

        producto = a * b * c;

        if (a < b && a < c) {
            if (b > c) {
                System.out.println("El número mayor es: " +b);
                System.out.println("El número menor es: " +a);
            } else {
                System.out.println("El número mayor es: "+c);
                System.out.println("El número menor es: "+a);
            if (b < c) {
                if (a > c) {
                    System.out.println("El número mayor es: " +a);
                    System.out.println("El número menor es: " +b);
                } else {
                    System.out.println("El número mayor es: "+c);
                    System.out.println("El número menor es: "+b);
                }
            } else {
                if (a > b) {
                        System.out.println("El número mayor es: " +a);
                        System.out.println("El número menor es: " +c);
                    } else {
                        System.out.println("El número mayor es: "+b);
                        System.out.println("El número menor es: "+c);
                    }
            }
        }

        System.out.println("El producto es: "+producto);
    }
}
```

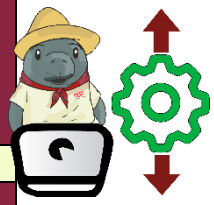
```
Problems  Intérprete de depuración  Consola
<terminado> Practica02_JAFPGV [Java Application] /Library/Java/JavaVirtualMachines/jdk-15.0.1.jdk/Contents/Home/bin/java  (24 jul. 2021 21:02:43 – 21:02:54)
Escribe el primer número:
10
Escribe el segundo número:
7
Escribe el tercer número:
5
El número mayor es: 10.0
El número menor es: 5.0
El producto es: 350.0
```


7. CONCLUSIONES: Al finalizar la práctica, con tus palabras contesta las siguientes preguntas:

¿Qué aprendí?

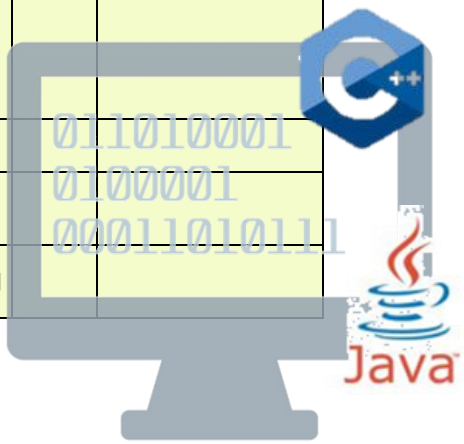
¿Dónde lo aplico en mi vida cotidiana y académica?

INSTRUMENTO DE EVALUACIÓN
LISTA DE COTEJO
Práctica 02. "Determinar de Tres Números Dados Cual es el Mayor, Cual el Menor y Realizar el Producto de Éstos"

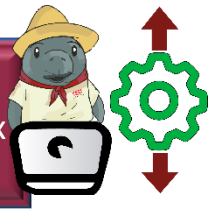


DATOS GENERALES	
Nombre(s) del alumno(s)	**Matricula(s)**
Producto: Programa de 3 Números Dados Cual es el Mayor, Cual el Menor y Realizar el Producto.	**Fecha**
Submódulo II: Programación en Java	**Periodo: 2026 – 2027A**
Nombre del docente	**Firma del docente**

CRITERIO	INDICADORES	VALOR OBTENIDO		PONDERACIÓN	CALIF	OBSERVACIONES Y/O SUGERENCIAS DE MEJORA
		SÍ	NO			
1	Codifica correctamente la secuencia lógica del desarrollo del programa.			**2**		
2	Identifica y aplica correctamente los métodos e instrucciones.			**2**		
3	Guarda el archivo con la nomenclatura solicitada.			**1**		
4	Compila y ejecuta el programa, mostrando en pantalla el resultado de determinar de 3 números cual es el mayor, cual el menor y su producto.			**3**		
5	Entrega en tiempo y forma.			**1**		
6	Presenta conclusiones de la práctica.			**1**		
	CALIFICACIÓN					



Estructura repetitiva FOR
Práctica No. 03 "Calcula la Función $e^x - x$, Considerando los Valores de x en el Intervalo Cerrado de -5 a 5"



PROPÓSITO: Identifica la estructura For y aplica las ventajas de utilizar la sentencia For dentro de los programas con sentencias repetitivas, para el desarrollo de programas en su vida escolar y profesional.

CONOCIMIENTOS

- ✓ Estructura For
- ✓ Variable inicial
- ✓ Variable final
- ✓ Condiciones(booleanas)
- ✓ Contador (incremento)

DESARROLLO

1. Realiza utilizando la estructura cíclica, un programa que calcule la función $e^x - x$, considerando los valores de x en el intervalo cerrado de -5 a 5.
2. Abre el editor del Lenguaje Java, Eclipse.
3. Guarda el proyecto o la clase en tu carpeta con la siguiente estructura: Practica04, siglas de tu nombre, grupo y turno. Ejemplo: "**Practica04_JAFPGV**" donde JAFP son las siglas de tu nombre, G es el grupo y V el turno para Matutino y V vespertino.
4. Escribe el código en el editor con apoyo de tu profesor.
5. Posteriormente compilamos el programa, presiona el botón de compilar  o utilizar la combinación de teclas **ctrl + espacio**.
6. A continuación, ejecutamos el programa, presiona el botón de ejecutar  o utilizar la combinación de teclas **ctrl + F11** se genera el archivo ejecutable de nuestro programa.

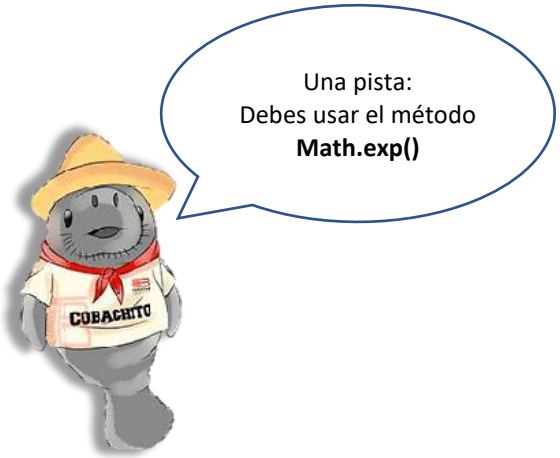

```
Practica02_JAFPGV.java   Practica03_JAFPGV.java
1  package primer_programa;
2
3  public class Practica03_JAFPGV {
4
5      public static void main(String[] args) {
6          // TODO Esbozo de método generado automáticamente
7
8          int c, x=-5;
9          double f=0;
10
11         for (c=1; c <= 11; c++)
12         {
13             System.out.print("El valor de x es: " +x);
14             f = Math.exp(x) - x;
15             System.out.println("   |  f(" +x +") = " +f);
16             x++;
17         }
18
19     }
20
21 }
22
```

```
Problems   Intérprete de depuración   Consola
<terminado> Practica03_JAFPGV [Java Application] /Library/Java/JavaVirtualMachines/jdk-15.0.1.jdk/Contents/Home/bin/java  (24 jul. 2021 22:57:16 – 22:57:19)
El valor de x es: -5   |  f(-5) = 5.006737946999086
El valor de x es: -4   |  f(-4) = 4.018315638888734
El valor de x es: -3   |  f(-3) = 3.049787068367864
El valor de x es: -2   |  f(-2) = 2.135335283236613
El valor de x es: -1   |  f(-1) = 1.3678794411714423
El valor de x es: 0   |  f(0) = 1.0
El valor de x es: 1   |  f(1) = 1.718281828459045
El valor de x es: 2   |  f(2) = 5.38905609893065
El valor de x es: 3   |  f(3) = 17.085536923187668
El valor de x es: 4   |  f(4) = 50.598150033144236
El valor de x es: 5   |  f(5) = 143.4131591025766
```

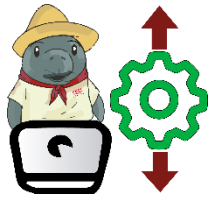
7. **CONCLUSIONES:** Al finalizar la práctica, con tus palabras contesta las siguientes preguntas:

¿Qué aprendí?

¿Dónde lo aplico en mi vida cotidiana y académica?

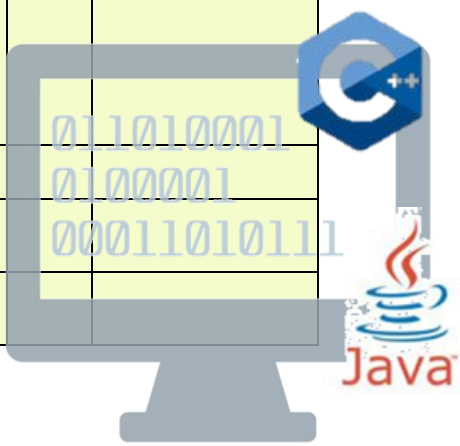


INSTRUMENTO DE EVALUACIÓN
LISTA DE COTEJO
Práctica 03. "Calcula la Función $e^x - x$, Considerando los Valores de x en el Intervalo Cerrado de -5 a 5"

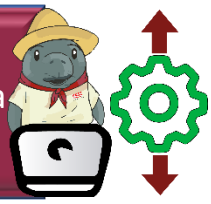


DATOS GENERALES	
Nombre(s) del alumno(s)	**Matricula(s)**
Producto: Calcula la función $e^x - x$, considerando los valores de x en el intervalo cerrado de -5 a 5.	**Fecha**
Submódulo II: Programación en Java	**Periodo: 2026 – 2027A**
Nombre del docente	**Firma del docente**

CRITERIO	INDICADORES	VALOR OBTENIDO		PONDERACIÓN	CALIF	OBSERVACIONES Y/O SUGERENCIAS DE MEJORA
		SÍ	NO			
1	Codifica correctamente la secuencia lógica del desarrollo del programa.			**2**		
2	Identifica y aplica correctamente los métodos e instrucciones.			**2**		
3	Guarda el archivo con la nomenclatura solicitada.			**1**		
4	Compila y ejecuta el programa, mostrando en pantalla el resultado del cálculo de la función $e^x - x$ considerando los valores de x en el intervalo cerrado de -5 a 5.			**3**		
5	Entrega en tiempo y forma.			**1**		
6	Presenta conclusiones de la práctica.			**1**		
				CALIFICACIÓN		



Estructura repetitiva WHILE
Práctica No. 04 "Realiza el Programa que Lea un Número e Imprima la Siguiente Serie: $\frac{1}{5}, \frac{3}{10}, \frac{9}{20}, \frac{27}{40}, \ldots$"



PROPÓSITO: Utiliza la estructura While para los programas con sentencias repetitivas y analiza sus ventajas y desventajas, teniendo en cuenta su utilidad en su vida escolar y profesional.

CONOCIMIENTOS

- ✓ Estructura While
- ✓ Variable inicial
- ✓ Variable final
- ✓ Condiciones(booleanas)
- ✓ Contador (incremento)

DESARROLLO

1. Realiza utilizando la estructura repetitiva, un programa que solicite un número cualquiera, e imprima la siguiente serie $\frac{1}{5}, \frac{3}{10}, \frac{9}{20}, \frac{27}{40}, \ldots$.
2. Abre el editor del Lenguaje Java, Eclipse.
3. Guarda el proyecto o la clase en tu carpeta con la siguiente estructura: Practica05, siglas de tu nombre, grupo y turno. Ejemplo: "**Practica05_JAFPGV**" donde JAFP son las siglas de tu nombre, G es el grupo y V el turno para Matutino y V vespertino.
4. Escribe el código en el editor con apoyo de tu profesor.
5. Posteriormente compilamos el programa, presiona el botón de compilar  o utilizar la combinación de teclas ctrl + espacio.
6. A continuación, ejecutamos el programa, presiona el botón de ejecutar  o utilizar la combinación de teclas ctrl + F11 se genera el archivo ejecutable de nuestro programa.

```java
package primer_programa;

import java.util.Scanner;

public class Practica04_JAFPGV {

    public static void main(String[] args) {
        // TODO Esbozo de método generado automáticamente

        int num = 1, den = 5, c = 1, n;

        Scanner lee = new Scanner(System.in);
        System.out.println("Escribe el número de la posición de la serie: ");

        n = lee.nextInt();

        while (n <= 0) {
            System.out.println("Debes escribir un número mayor o igual a 1");
            n = lee.nextInt();
        }


        while (c <= n)
        {
            System.out.print(+num +"/" +den +", ");
            num = num * 3;
            den = den * 2;
            c++;
        }
    }
}
```

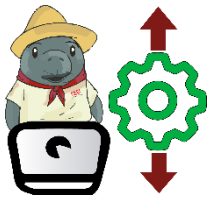
```
<terminado> Practica04_JAFPGV [Java Application] /Library/Java/JavaVirtualMachines/jdk-15.0.1.jdk/Contents/Home/bin/java (26 jul. 2021 10:31:48 – 10:31:59)
Escribe el número de la posición de la serie:
7
1/5, 3/10, 9/20, 27/40, 81/80, 243/160, 729/320,
```

7. **CONCLUSIONES:** Al finalizar la práctica, con tus palabras contesta las siguientes preguntas:

¿Qué aprendí?

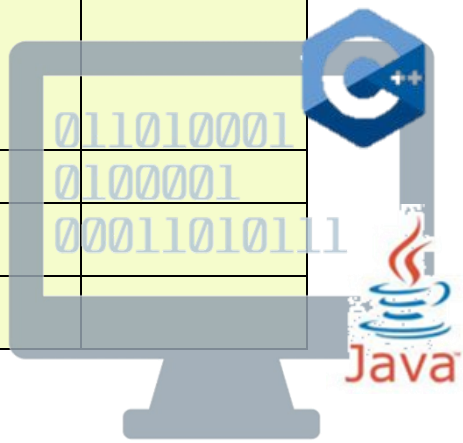
¿Dónde lo aplico en mi vida cotidiana y académica?

INSTRUMENTO DE EVALUACIÓN
LISTA DE COTEJO
Práctica 04. "Realiza el Programa que Lea un Número e Imprima la Siguiente Serie: $\frac{1}{5}, \frac{3}{10}, \frac{9}{20}, \frac{27}{40}, \ldots$"



DATOS GENERALES	
Nombre(s) del alumno(s)	**Matricula(s)**
Producto: Programa que Solicite un Número Cualquiera, e Imprima la Siguiente Serie $\frac{1}{5}, \frac{3}{10}, \frac{9}{20}, \frac{27}{40}, \ldots.$	**Fecha**
Submódulo II: Programación en Java	**Periodo: 2026 – 2027A**
Nombre del docente	**Firma del docente**

CRITERIO	INDICADORES	VALOR OBTENIDO		PONDERACIÓN	CALIF	OBSERVACIONES Y/O SUGERENCIAS DE MEJORA
		SÍ	NO			
1	Codifica correctamente la secuencia lógica del desarrollo del programa.			**2**		
2	Identifica y aplica correctamente los métodos e instrucciones.			**2**		
3	Guarda el archivo con la nomenclatura solicitada.			**1**		
4	Compila y ejecuta el programa un programa que solicite un número cualquiera, e imprima la siguiente serie $\frac{1}{5}, \frac{3}{10}, \frac{9}{20}, \frac{27}{40}, \ldots.$			**3**		
5	Entrega en tiempo y forma.			**1**		
6	Presenta conclusiones de la práctica.			**1**		
				CALIFICACIÓN		



Estructura repetitiva DO-WHILE
Practica No. 05 "Determinar en un Conjunto de n Números Naturales"

PROPÓSITO: Utiliza el ciclo DO … WHILE para la resolución de problemas en los programas, teniendo en cuenta su utilidad tanto en su vida escolar y profesional.

CONOCIMIENTOS

- ✓ Estructura condicional if
- ✓ Estructura repetitiva Do.While
- ✓ Ciclos Anidados
- ✓ Variable inicial
- ✓ Variable final
- ✓ Condiciones(booleanas)
- ✓ Contador (incremento)

DESARROLLO

1. Realiza utilizando la estructura repetitiva, un programa que determina en un conjunto de n números naturales:

   ¿Cuántos son menores que 15? ¿Cuántos son mayores que 50?

   ¿Cuántos están en el rango entre 25 y 45?

2. Abre el editor del Lenguaje Java, Eclipse.
3. Guarda el proyecto o la clase en tu carpeta con la siguiente estructura: Practica06, siglas de tu nombre, grupo y turno. Ejemplo: "**Practica06_JAFPGV**" donde JAFP son las siglas de tu nombre, G es el grupo y V el turno para Matutino y V vespertino.
4. Escribe el código en el editor con apoyo de tu profesor.

5. Posteriormente compilamos el programa, presiona el botón de compilar  o utilizar la combinación de teclas ctrl + espacio.
6. A continuación, ejecutamos el programa, presiona el botón de ejecutar  o utilizar la combinación de teclas ctrl + F11 se genera el archivo ejecutable de nuestro programa.


```
*Practica05_JAFPGV.java
1  package primer_programa;
2
3  import java.util.Scanner;
4
5  public class Practica05_JAFPGV {
6
7      public static void main(String[] args) {
8          // TODO Esbozo de método generado automáticamente
9
10         int n, num;
11
12         int c = 0, c15 = 0, c50 = 0, c_rango = 0;
13
14         Scanner lee = new Scanner (System.in);
15         System.out.print("Ingresa la cantidad de números: ");
16         n = lee.nextInt();
17
18         System.out.println("");
19         do {
20             System.out.print(+(c+1) +".- Ingresar numero: ");
21             num = lee.nextInt();
22
23             if (num < 15)
24                 c15++;
25
26             if (num > 50)
27                 c50++;
28             if (num >= 25 && num <= 45)
29                 c_rango++;
30
31             c++;
32
33         } while (c < n);
34
35         System.out.println("Menores a 15: " +c15);
36         System.out.println("Mayores a 50: " +c50);
37         System.out.println("Entre 25 y 45: " +c_rango);
38     }
39
40 }
41
```

```
Problems  Intérprete de depuración  Consola
<terminado> Practica05_JAFPGV [Java Application] /Library/Java/JavaVirtualMachines/jdk-15.0.1.jdk/Contents/Home/bin/java (26 jul. 2021 11:15:52 – 11:17:07)
Ingresa la cantidad de números: 7

1.- Ingresar numero: 12
2.- Ingresar numero: 7
3.- Ingresar numero: 27
4.- Ingresar numero: 31
5.- Ingresar numero: 33
6.- Ingresar numero: 57
7.- Ingresar numero: 60
Menores a 15: 2Mayores a 50: 2Entre 25 y 45: 3
```

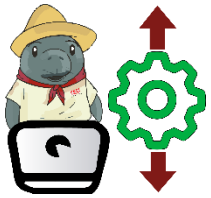
7. **CONCLUSIONES:** Al finalizar la práctica, con tus palabras contesta las siguientes preguntas:

Te ayudo:
Debes usar if e incrementos dentro del ciclo do … while.
¡Éxito!

¿Qué aprendí?

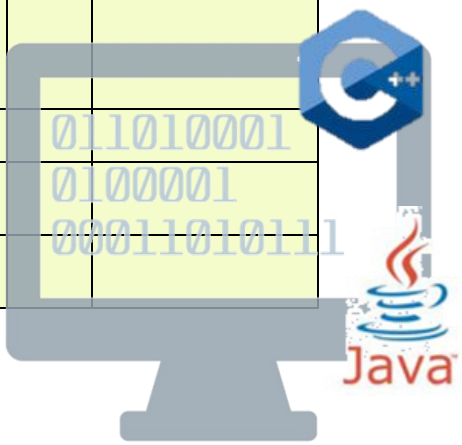
¿Dónde lo aplico en mi vida cotidiana y académica?

INSTRUMENTO DE EVALUACIÓN
LISTA DE COTEJO
Practica 05. "Determinar en un Conjunto de n Números Naturales"



DATOS GENERALES		
Nombre(s) del alumno(s)		**Matricula(s)**
Producto: Programa que Determina en un Conjunto de n Números Naturales.		**Fecha**
Submódulo II: Programación en Java		**Periodo: 2026 – 2027A**
Nombre del docente		**Firma del docente**

CRITERIO	INDICADORES	VALOR OBTENIDO		PONDERACIÓN	CALIF	OBSERVACIONES Y/O SUGERENCIAS DE MEJORA
		SÍ	NO			
1	Codifica correctamente la secuencia lógica del desarrollo del programa.			**2**		
2	Identifica y aplica correctamente las librerías e instrucciones.			**2**		
3	Guarda el archivo con la nomenclatura solicitada.			**1**		
4	Compila y ejecuta el programa que determina en un conjunto de n números naturales: Cuántos son menores que 15. Cuántos son mayores que 50. Cuántos están en el rango entre 25 y 45.			**3**		
5	Entrega en tiempo y forma.			**1**		
6	Presenta conclusiones de la práctica.			**1**		
				CALIFICACIÓN		



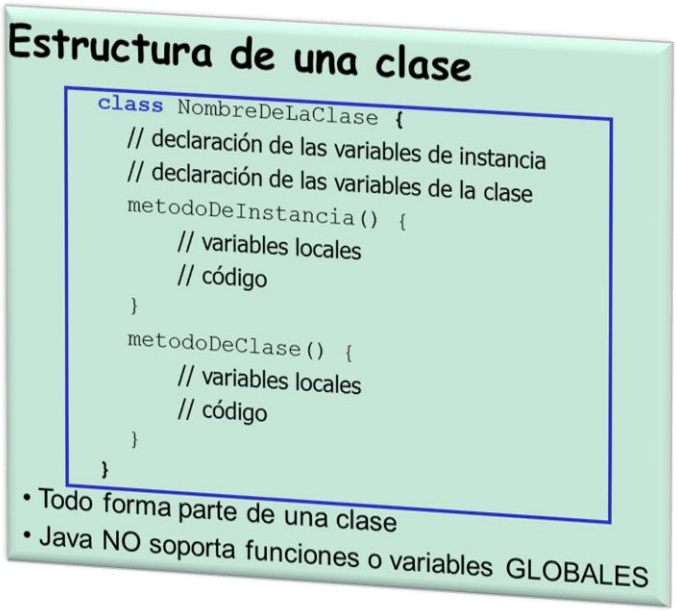
Lectura No. 6 "Clases y Objetos"

Instrucciones: *Realiza la Lectura 6. "Clases y Objetos" subrayando las ideas principales, y al finalizar realiza la Práctica 6 – Clases y Objetos*

LECTURA 6. "CLASES Y OBJETOS"

La **programación orientada a objetos** se enfoca en los elementos de un sistema, sus atributos y las interacciones que se producen entre ellos para diseñar aplicaciones informáticas. Los elementos abstractos del modelo orientado a objetos se denominan ***clases***.

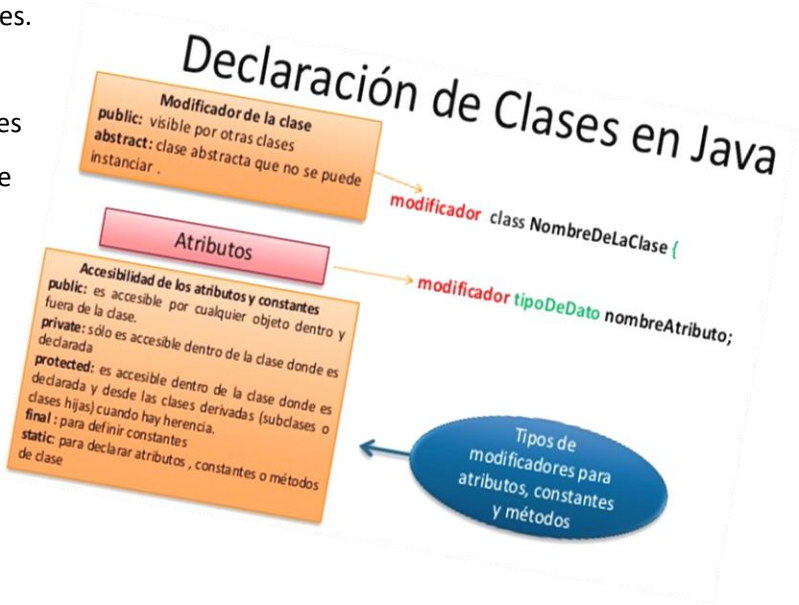
Un programa Java utiliza **clases y objetos**. Las **clases** representan un esquema simplificado de la casuística de una aplicación informática. Una **clase** es una representación abstracta de un conjunto de objetos que comparten los mismos atributos y comportamiento, es decir, una clase describe un tipo de objetos. Un **objeto** es una instancia de una clase, tiene una identidad propia y un estado. La identidad de un objeto se define por su identificador. El estado de un objeto se define por el valor de sus **atributos**. El comportamiento de un objeto queda determinado por el comportamiento de la clase a la que pertenece. Los objetos son unidades indivisibles y disponen de mecanismos de interacción llamados **métodos**.



De este modo, *una clase es un template (un modelo) para un objeto, y un objeto es una instancia de una clase*. Debido a que un objeto es una instancia de una clase, a menudo las dos palabras objeto e instancia se usan indistintamente.

Por ejemplo, si se desea diseñar un programa en Java para gestionar las ventas de una tienda, entonces habría que identificar y describir las características de elementos como: cliente, tipo de cliente, producto, pedido, tipo de entrega, forma de pago, unidades en existencia de los productos, etc. Los procesos de gestión de la tienda incluirían el registro de los clientes, el registro de los productos de la tienda, el proceso de compra del cliente y la realización de los pedidos, la entrada y salida de productos del almacén, etc. Si en vez de una tienda se trata de una aplicación para dibujar figuras geométricas en dos dimensiones, entonces sería necesario identificar y describir las características de las figuras y su posición. En este caso habría figuras de tipo círculo, rectángulo, triángulo, etc. Las operaciones para realizar con las figuras incluirían dibujar, borrar, mover o rotar.

En su forma más simple, una clase se define por la palabra reservada class seguida del nombre de la clase. *El nombre de la clase debe empezar por mayúscula*. Si el nombre es compuesto, entonces cada palabra debe empezar por mayúscula. Ejemplo: Circulo, Rectangulo, Triangulo y FiguraGeometrica son nombres válidos de clases.



Por ejemplo, la clase Circulo se define con tres atributos: el radio y las coordenadas x, y que definen la posición del centro del círculo.

```
/* Esta clase define los atributos
      de un círculo */

public class Circulo {

int x;
int y;
int radio;

}
```

Una vez que se ha declarado una clase, se pueden crear objetos a partir de ella. A la creación de un objeto se le denomina **instanciación**. Es por esto que se dice que un objeto es una instancia de una clase y el término instancia y objeto se utilizan indistintamente.

Para crear objetos, basta con declarar una variable de alguno de los tipos de figuras geométricas:

```
Circulo circulo1;
Circulo circulo2;
```

Para crear el objeto y asignar un espacio de memoria es necesario realizar la instanciación con el operador new.

```
circulo1 = new Circulo();
circulo2 = new Circulo();
```

Después de crear los objetos, circulo1 y circulo2 almacenan los valores predeterminados de la clase Circulo. A partir de este momento los objetos ya pueden ser referenciados por su nombre. Los nombres circulo1 y circulo2 son las referencias válidas para utilizar ambos objetos.

Los elementos de una clase

Una clase describe un tipo de objetos con características comunes. Es **necesario definir la información que almacena el objeto y su comportamiento**.

Clase Cuenta en Java

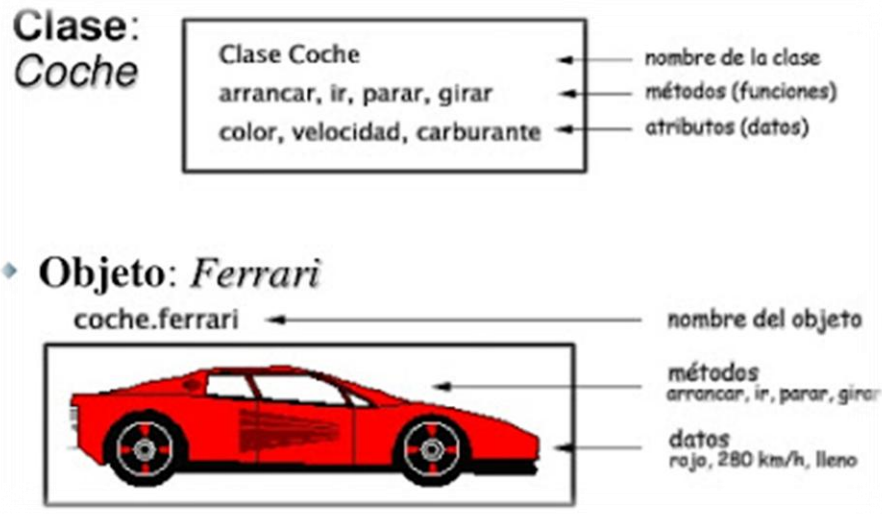
```
class Cuenta{
    String titular;
    double saldo;                          ATRIBUTOS

    void ingreso (double cantidad){
        saldo = saldo + cantidad;
    }
    void reintegro (double cantidad){
        if (cantidad <= saldo)
            saldo = saldo - cantidad;
    }                                      MÉTODOS
}
```

►ATRIBUTOS

La información de un objeto se almacena en *atributos*. Los atributos pueden ser de tipos primitivos de Java o de tipo objeto.

Por ejemplo: Para el catálogo de vehículos de una empresa de alquiler, es necesario conocer la matrícula del coche, su marca, modelo, color, la tarifa del alquiler y su disponibilidad.

```
public class Vehiculo {
String matricula;
String marca;
String modelo;
String color;
double tarifa;
boolean disponible;
}
```



En este ejemplo, los atributos matricula, marca, modelo y color son cadenas de caracteres, tarifa es un número real y disponible es un valor lógico.

Métodos y constructores

Además de definir los atributos de un objeto, es necesario definir los métodos que determinan su comportamiento. Toda clase debe definir un método especial denominado constructor para instanciar los objetos de la clase. Este método tiene el mismo nombre de la clase. Por ejemplo, para la clase Vehiculo, el identificador del método constructor es Vehiculo. El método constructor se ejecuta cada vez que se instancia un objeto de la clase. Este método se utiliza para inicializar los atributos del objeto que se instancia.

Para diferenciar entre los atributos del objeto y los identificadores de los parámetros del método constructor, se utiliza la palabra this. De esta forma, los parámetros del método pueden tener el mismo nombre que los atributos de la clase. Esto permite hacer una asignación como la que se muestra a continuación, donde this.marca se refiere al atributo del objeto y marca al parámetro del método.

this.marca = marca;

ejemplo.

```
public class Vehiculo {
        String matricula;
        String marca;
        String modelo;
        String color;
        double tarifa;
        boolean disponible;

// el método constructor de la clase Vehiculo
        public Vehiculo(String matricula,
                String marca,
                String modelo,
                String color,
                double tarifa) {
        this,matricula = matricula;
        this.marca = marca;
        this.modelo = modelo;
        this.color = color;
        this.tarifa = tarifa;
        this.disponible = false;
        }
}
```

Para acceder a los atributos de los objetos de la clase Vehiculo se definen los métodos 'get' y 'set'. Los métodos '**get**' se utilizan para consultar el estado de un objeto y los métodos '**set**' para modificar su estado.

En la clase Vehiculo es necesario definir un método 'get' para cada uno de sus atributos:

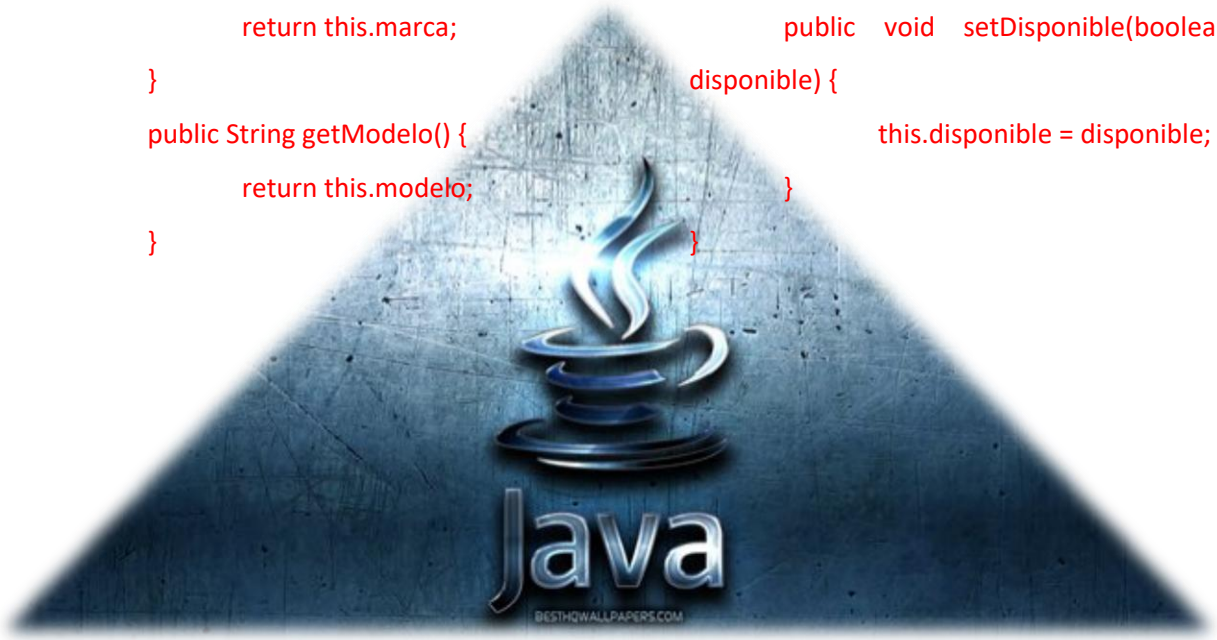
getMatricula(), getMarca(), getModelo(), getColor(), getTarifa() y getDisponible().

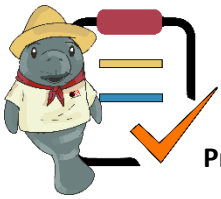
Los métodos 'set' solo se definen para los atributos que pueden ser modificados después de que se ha creado el objeto.

En este caso es necesario definir setTarifa (double tarifa) y setDisponible (boolean disponible) para modificar la tarifa del alquiler del vehículo y su disponibilidad, respectivamente.

Ejemplo:

```
public class Vehiculo {
        String matricula;
        String marca;
        String modelo;
        String color;
        double tarifa;
        boolean disponible;
// los métodos 'get' y 'set' de la clase Vehiculo
        public String getMatricula() {
                return this.matricula;
        }
        public String getMarca() {
                return this.marca;
        }
        public String getModelo() {
                return this.modelo;
        }
        public String getColor() {
                return this.color;
        }
        public double getTarifa() {
                return this.tarifa;
        }
        public boolean getDisponible() {
                return this.disponible;
        }
        public void setTarifa(double tarifa) {
                this.tarifa = tarifa;
        }
        public void setDisponible(boolean disponible) {
                this.disponible = disponible;
        }
}
```





Actividad 05 "Ordena el Código Feliz Día"

Programa: Lee un par de fechas, la fecha de nacimiento del usuario y la fecha actual, al final escribe un mensaje en pantalla para felicitar o expresar ¡Feliz Día!

1. Lee y analiza el código de **clases y objetos** en lenguaje de Java dentro de los rectángulos, y con el apoyo de tu maestro, enumera los círculos de cada rectángulo en el orden lógico que tendría el programa.
2. A continuación, escribe y comprueba su funcionamiento dentro del editor de Eclipse (compílalo y ejecútalo).


```
Problems  Intérprete de depuración  Consola
<terminado> Cumple [Java Application] /Library/Java/JavaVirtualMachines/jdk-15.0.1.jdk/Contents/Home/bin/java  (26 jul. 2021 12:58:32 – 12:59:11)
Introduzca la fecha de hoy:
El día: 22
El mes: 08
Introduzca su fecha de nacimiento:
El día: 01
El mes: 10
La fecha de hoy es
La fecha es: 22/ 8
Su fecha de nacimiento es
La fecha es: 1/ 10
¡Feliz día!
```

```
Problems  Intérprete de depuración  Consola
<terminado> Cumple [Java Application] /Library/Java/JavaVirtualMachines/jdk-15.0.1.jdk/Contents/Home/bin/java  (26 jul. 2021 12:59:56 – 13:02:29)
Introduzca la fecha de hoy:
El día: 06
El mes: 12
Introduzca su fecha de nacimiento:
El día: 06
El mes: 12
La fecha de hoy es
La fecha es: 6/ 12
Su fecha de nacimiento es
La fecha es: 6/ 12
¡Muy Feliz Cumpleaños!
```

3. Debes crear dos clases y solamente la clase Cumple tendrá la función principal main.
4. **Señala dónde están las clases de color rojo y los objetos de color azul.**

5. Utilizando la inteligencia artificial que tengas en tu dispositivo, solicita ejemplos de clases y objetos para ser codificados en el IDE de Java.

DiaAnyo.java

CLASE DiaAnyo

Enumera del 1 al 4

```
    public void visualizar ()
    {
        System.out.println("La fecha es: " +dia +"/ " +mes);
    }
}
```

```
package primer_programa;

public class DiaAnyo {

    private final int dia;
    private final int mes;
```

```
public boolean igual (DiaAnyo d)
{
    if (dia == d.dia && mes == d.mes)
        return true;
    else
        return false;

}
```

```
public DiaAnyo (int d, int m)
{
    dia = d;
    mes = m;
}
```

Cumple.java

CLASE Cumple

Enumera del 1 al 6


```
        System.out.println ("Introduzca su fecha de nacimiento:");
        System.out.print ("El día: ");
        d = entrada.nextInt();

        System.out.print ("El mes: ");
        m = entrada.nextInt();

        cumpleanyos = new DiaAnyo (d, m);
```

```
        if (hoy.igual(cumpleanyos))
            System.out.println("¡Muy Feliz Cumpleaños!");
        else
            System.out.println("¡Feliz día!");
    }

}
```

```
public class Cumple {

    public static void main (String[] arg) throws IOException {

        DiaAnyo hoy;
        DiaAnyo cumpleanyos;

        int d, m;
```

```
        Scanner entrada = new Scanner (System.in);

        System.out.println ("Introduzca la fecha de hoy:");
        System.out.print ("El día: ");
        d = entrada.nextInt();

        System.out.print ("El mes: ");
        m = entrada.nextInt();

        hoy = new DiaAnyo (d, m);
```

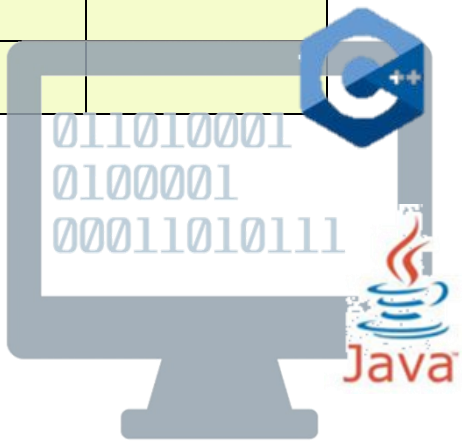
```
        System.out.println("La fecha de hoy es ");
        hoy.visualizar();

        System.out.println("Su fecha de nacimiento es ");
        cumpleanyos.visualizar();
```

```
package primer_programa;

import java.io.*;
import java.util.*;
```

INSTRUMENTO DE EVALUACIÓN
LISTA DE COTEJO
Actividad 05. Enumera el Código Feliz Día.

DATOS GENERALES	
Nombre(s) del alumno(s)	**Matricula(s)**
Producto: Enumeración y Escritura del Código.	**Fecha**
Submódulo II: Programación en Java	**Periodo: 2025 – 2026A**
Nombre del docente	**Firma del docente**

CRITERIO	INDICADORES	VALOR OBTENIDO		PONDERACIÓN	CALIF	OBSERVACIONES Y/O SUGERENCIAS DE MEJORA
		SÍ	NO			
1	Entrego en tiempo y Forma.			**2**		
2	Identifica correctamente las clases y los objeto en el código del programa.			**2**		
3	Se aprecia el conocimiento obtenido de la aplicación.			**1**		
4	Enumera y escribe el código.			**2**		
5	Utiliza la tecnología para desarrollar la actividad.			**2**		
6	Mantienen interés en la comprensión de la actividad.			**1**		
				CALIFICACIÓN		



Programa:

DiaAnyo.java | Cumple.java

```
1  package primer_programa;
2
3  public class DiaAnyo {
4
5      private final int dia;
6      private final int mes;
7
8      public DiaAnyo (int d, int m)
9      {
10         dia = d;
11         mes = m;
12     }
13
14     public boolean igual (DiaAnyo d)
15     {
16         if (dia == d.dia && mes == d.mes)
17             return true;
18         else
19             return false;
20
21     }
22
23     public void visualizar ()
24     {
25         System.out.println("La fecha es: " +dia +"/ " +mes);
26     }
27 }
28
```

DiaAnyo.java | Cumple.java

```java
package primer_programa;

import java.io.*;
import java.util.*;

public class Cumple {

    public static void main (String[] arg) throws IOException {

        DiaAnyo hoy;
        DiaAnyo cumpleanyos;

        int d, m;

        Scanner entrada = new Scanner (System.in);

        System.out.println ("Introduzca la fecha de hoy:");
        System.out.print ("El día: ");
        d = entrada.nextInt();

        System.out.print ("El mes: ");
        m = entrada.nextInt();

        hoy = new DiaAnyo (d, m);


        System.out.println ("Introduzca su fecha de nacimiento:");
        System.out.print ("El día: ");
        d = entrada.nextInt();

        System.out.print ("El mes: ");
        m = entrada.nextInt();

        cumpleanyos = new DiaAnyo (d, m);


        System.out.println("La fecha de hoy es ");
        hoy.visualizar();

        System.out.println("Su fecha de nacimiento es ");
        cumpleanyos.visualizar();

```


```java

        if (hoy.igual(cumpleanyos))
            System.out.println("¡Muy Feliz Cumpleaños!");
        else
            System.out.println("¡Feliz día!");
    }


}
```

CLASES Y OBJETOS
Práctica No. 06 "Registra Tu Asistencia"

PROPÓSITO: Distinguir las instrucciones que son necesarias para la realización de programas en Java, aplicando clases y objetos en el desarrollo de programas.

CONOCIMIENTOS

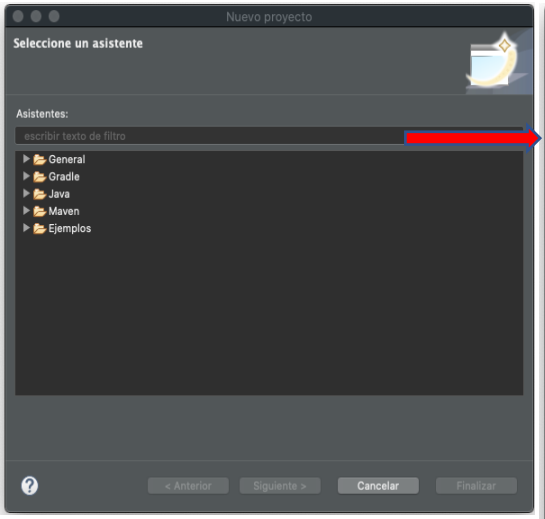
- ✓ Proyectos
- ✓ Clases
- ✓ Constructores
- ✓ Método de lectura
- ✓ Método de escritura
- ✓ Objetos
- ✓ Funciones

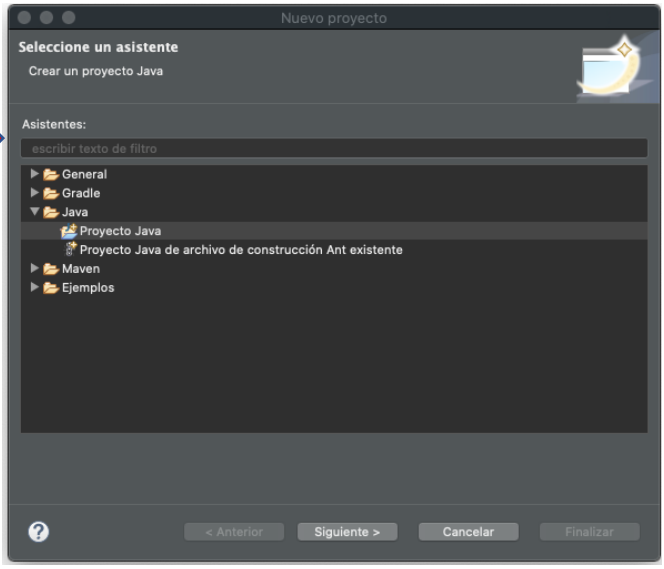
DESARROLLO

1. Realiza un programa en el que un estudiante pueda registrar su asistencia y salida en el lugar al que ingresa en el plantel. Debes considerar:
   - La Clase Estudiante (Atributos: Matrícula, Nombre Completo, Semestre, Grupo, Capacitación, Turno, Plantel; y sus Métodos: Constructor (), Visualizar ()).
   - La Clase Lugar: (Atributos: Espacio, Hora, Minutos, Día, Mes; y sus Métodos: Constructor (), Ingreso (), Salida ()).
   - La Clase Principal (Registro)
2. Abre el editor del Lenguaje Java, Eclipse.

3. Ahora abre el menú **Archivo** y elige la opción **Nuevo** para crear un proyecto, o usa el ícono 

4. Usa el asistente para elegir **Java**:

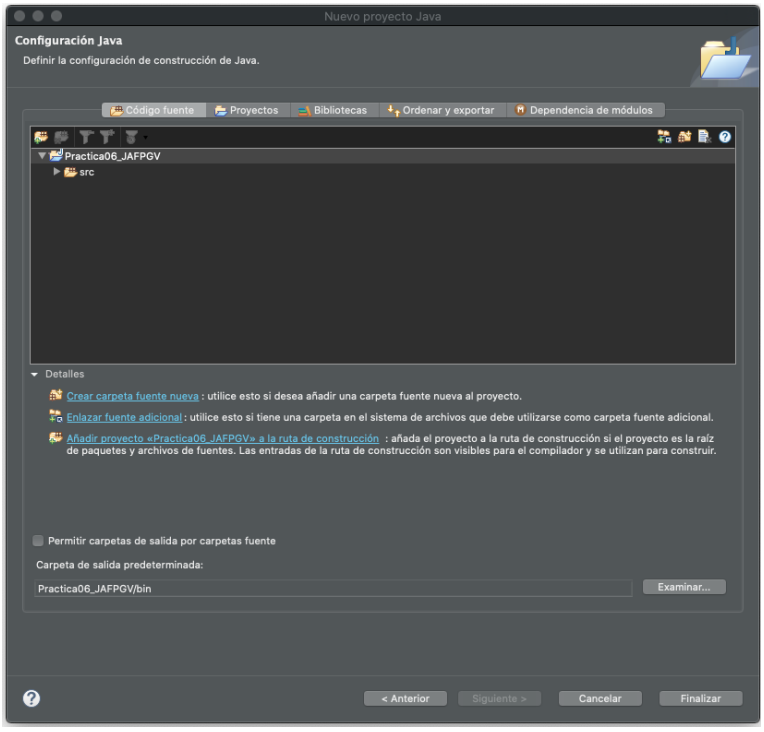




5. Escribe el nombre del proyecto con la siguiente estructura: Practica01, siglas de tu nombre, grupo y turno. Ejemplo: "**Practica01_JAFPGV**" donde JAFP son las siglas de tu nombre, G es el grupo y V el turno para Matutino y V vespertino.

6. Guarda el proyecto o la clase en tu carpeta.

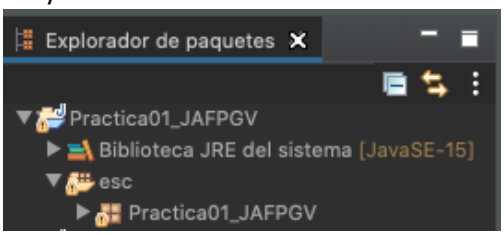
7. Ahora da clic derecho al proyecto o elige nuevamente el botón **nuevo**.



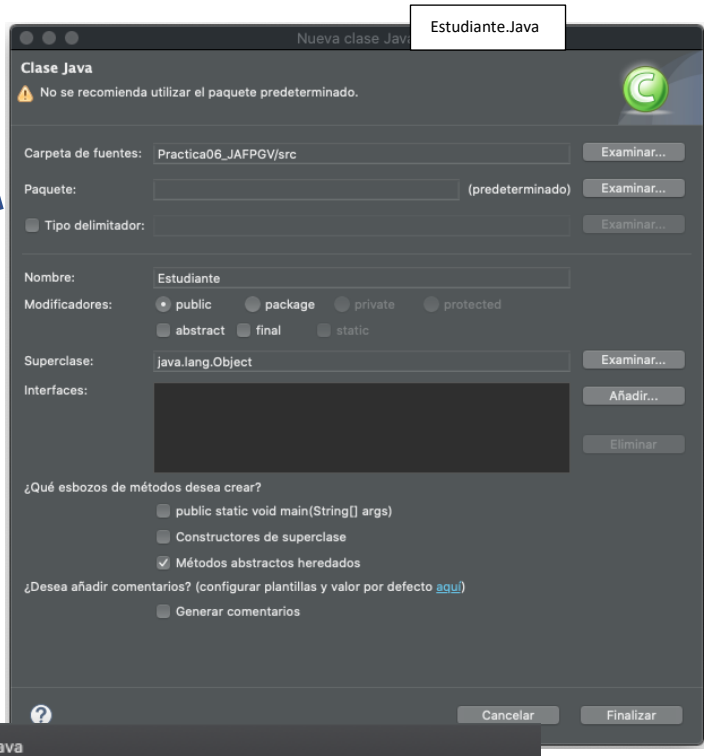
8. Elige Clase o usa el ícono  .

9. Dale el nombre **Estudiante**.

Asegúrate que el proyecto quede de la siguiente manera en el panel de Proyectos:



10. Escribe el código:



Estudiante.java | Justificado.java | Registro.java

```java
package Practica01_JAFPGV;

public class Estudiante {

    private final String matricula;
    private final String nombre;
    private final int semestre;
    private final char grupo;
    private final String capacitacion;
    private final String turno;
    private final String plantel;


    public Estudiante (String m, String n, int s, char g, String c, String t, String p)
    {
        matricula = m;
        nombre = n;
        semestre = s;
        grupo = g;
        capacitacion = c;
        turno = t;
        plantel = p;
    }

    void visualizar ()
    {
        System.out.println("Estudiante: " +nombre);
        System.out.println(matricula);
        System.out.println(+semestre +" " +grupo);
        System.out.println("Capacitación: " +capacitacion);
        System.out.println("Turno: " +turno);
        System.out.println("Plantel Nº" +plantel);
    }
}

```

11. Guarda el código  .

12. Realiza el mismo procedimiento a partir del punto 8 hasta el 11, para realizar la clase Lugar, (en el ejemplo se llama: Justificado).

Estudiante.java | Justificado.java | Registro.java

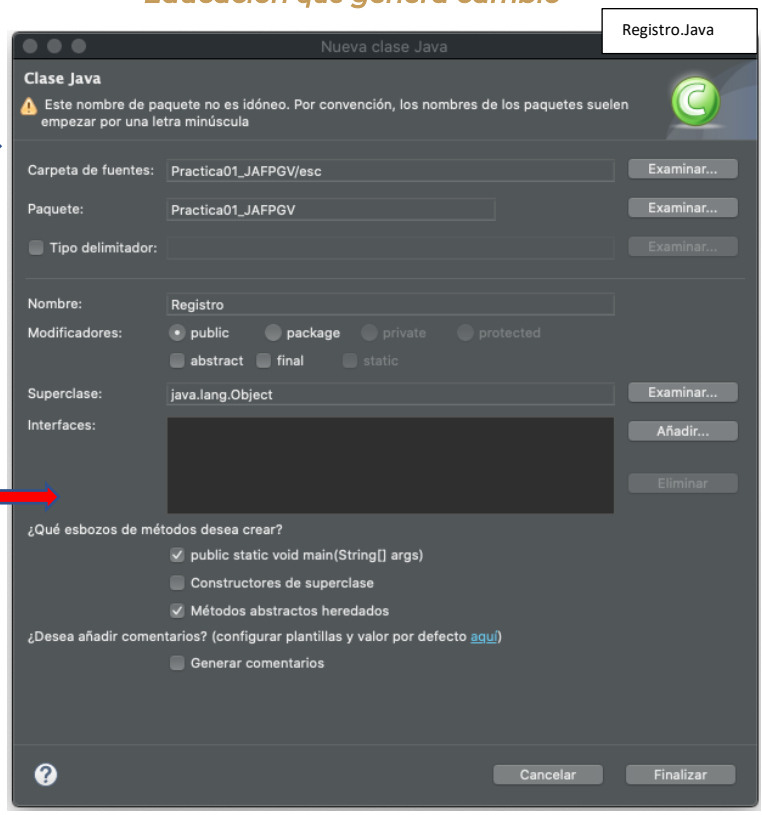
```
1  package Practica01_JAFPGV;
2
3  public class Justificado {
4
5      private final String espacio;
6      private final int hora;
7      private final int minutos;
8      private final int dia;
9      private final int mes;
10
11
12     public Justificado (String e, int h, int min, int d, int m)
13     {
14         espacio = e;
15         hora = h;
16         minutos = min;
17         dia = d;
18         mes = m;
19     }
20
21     public void Ingreso (String n, Justificado e)
22     {
23         System.out.println("");
24         System.out.println("INGRESO   FECHA " +e.dia +"/" +e.mes +"   HORA " +e.hora +":" +e.minutos);
25     }
26
27     public void Salida (String n, int h, int m, Justificado e)
28     {
29         System.out.println("");
30         System.out.println("");
31         System.out.println("Salida registrada de " +e.espacio);
32         System.out.println("Hasta pronto " +n);
33         System.out.println("FECHA " +e.dia +"/" +e.mes +"   HORA " +h +":" +m);
34     }
35 }
36
```

13. Finalmente elige Clase o usa el ícono  .

14. Dale el nombre **Registro**.

15. Activa la casilla **public void main(String[] args)** para que incluya el método principal.

16. Captura el código siguiente:



Estudiante.java | Justificado.java | Registro.java

```java
package Practica01_JAFPGV;

import java.util.Scanner;

public class Registro {

    public static void main(String[] args) {
        // Función principal

        String m, n, c, t, p, e, r;
        char g, sal;
        int s, h, min, d, mes;


        Scanner entrada = new Scanner (System.in);

        Estudiante est;
        Justificado lugar;

        System.out.println("Registro: ");

        System.out.println("");
        System.out.print("Matricula: ");
        m = entrada.nextLine();

        System.out.println("");
        System.out.print("Nombre: ");
        n = entrada.nextLine();

        System.out.println("");
        System.out.print("Semestre (Número): ");
        s = entrada.nextInt();

        System.out.println("");
        System.out.print("Grupo: ");
        g = entrada.next().charAt(0);

        r = entrada.nextLine();

        System.out.println("");
        System.out.print("Capacitación: ");
        c = entrada.nextLine();

        System.out.println("");
        System.out.print("Turno: ");
        t = entrada.nextLine();

        System.out.println("");
        System.out.print("Plantel: ");
        p = entrada.nextLine();

        est = new Estudiante(m, n, s, g, c, t, p);


        System.out.println("");
        System.out.print("Asistencia en: ");
        e = entrada.nextLine();

        System.out.println("");
        System.out.print("Hora: ");
        h = entrada.nextInt();

        System.out.println("");
        System.out.print("Minutos: ");
        min = entrada.nextInt();

        System.out.println("");
        System.out.print("Día: ");
        d = entrada.nextInt();

        System.out.println("");
        System.out.print("Mes: ");
        mes = entrada.nextInt();

        lugar = new Justificado (e, h, min, d, mes);

        lugar.Ingreso(n, lugar);

```

```
80
81          System.out.println("");
82          System.out.println("¿Deseas registrar tu salida");
83          sal = entrada.next().charAt(0);
84
85          if (sal == 's' || sal == 'S')
86              System.out.println("");
87              System.out.print("Hora: ");
88              h = entrada.nextInt();
89
90              System.out.println("");
91              System.out.print("Minutos: ");
92              min = entrada.nextInt();
93
94              lugar.Salida(n, h, min, lugar);
95
96              System.out.println("");
97              est.visualizar();
98
99
100     }
101
102 }
103
```

17. Posteriormente compilamos el programa, presiona el botón de compilar  o utilizar la combinación de tecla ctrl + espacio.

18. A continuación, ejecutamos el programa, presiona el botón de ejecutar  o utilizar la combinación de teclas ctrl + F11 se genera el archivo ejecutable de nuestro programa.


```
Problems  Javadoc  Declaration  Consola
<terminado> Registro [Java Application] /Library/Java/JavaVirtualMachines/jdk-15.0.1.jdk/Contents/Home/bin/java  (27 jul. 2021 8:47:07 – 8:48:45)
Registro:

Matricula: 04B-002949

Nombre: Jorge Armando Flores Palacios

Semestre (Número): 5

Grupo: G

Capacitación: Desarrollo de Software

Turno: Vespertino

Plantel: 06 "Cunduacán"

Asistencia en: Biblioteca

Hora: 14

Minutos: 10

Día: 27

Mes: 07

INGRESO   FECHA 27/7   HORA 14:10

¿Deseas registrar tu salida
s

Hora: 15

Minutos: 00


Hora: 15

Minutos: 00


Salida registrada de Biblioteca
Hasta pronto Jorge Armando Flores Palacios
FECHA 27/7   HORA 15:0

Estudiante: Jorge Armando Flores Palacios
04B-002949
5 G
Capacitación: Desarrollo de Software
Turno: Vespertino
Plantel Nº06 "Cunduacán"
```

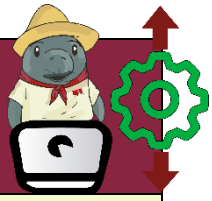
CONCLUSIONES:

Al finalizar la práctica, con tus palabras contesta las siguientes preguntas:

¿Qué aprendí?

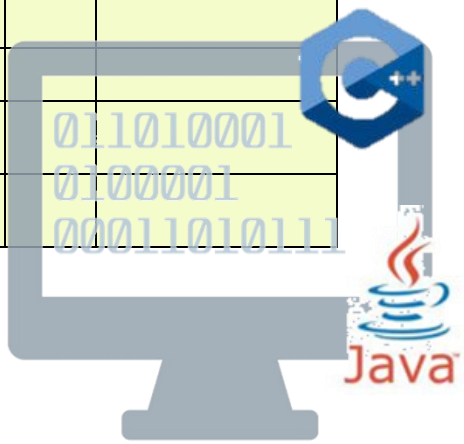
¿Dónde lo aplico en mi vida cotidiana y académica?

INSTRUMENTO DE EVALUACIÓN
LISTA DE COTEJO
Practica 06. "Registra tu Asistencia"



DATOS GENERALES		
Nombre(s) del alumno(s)		**Matricula(s)**
Producto: Programa que Registra los Datos de un Estudiante y Su Asistencia en un Lugar.		**Fecha**
Submódulo II: Programación en Java		**Periodo: 2026 – 2027A**
Nombre del docente		**Firma del docente**

CRITERIO	INDICADORES	VALOR OBTENIDO		PONDERACIÓN	CALIF	OBSERVACIONES Y/O SUGERENCIAS DE MEJORA
		SÍ	NO			
1	Codifica correctamente la secuencia lógica del desarrollo del programa.			**2**		
2	Identifica y aplica correctamente las Clases e instrucciones.			**2**		
3	Guarda el archivo con la nomenclatura solicitada.			**1**		
4	Compila y ejecuta el programa que registra la asistencia de un estudiante en un lugar del colegio.			**3**		
5	Entrega en tiempo y forma.			**1**		
6	Presenta conclusiones de la práctica.			**1**		
				CALIFICACIÓN		

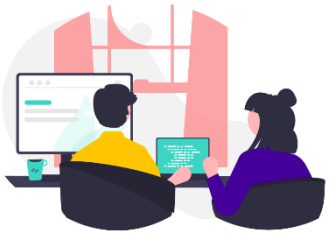


SIGA

REDUCE TU HUELLA

Propósito:

Desarrollar un programa en JAVA empleando un IDE para su compilación y ejecución, dentro de un equipo de 6 integrantes; que permita calcular la cantidad aproximada e CO2 expresada en Kilogramos (KgCO2e) recopilando datos de sus actividades diarias, utilizando los parámetros de factores de emisión publicados por las dependencias oficiales de México y posteriormente mostrar el resultado junto con las mejores recomendaciones para reducir la huella de carbono en el aula o en un espacio virtual.



Instrucciones: *Con los conocimientos que has adquirido a lo largo del submódulo, es momento de diseñar en equipo, un programa en el que se hace conciencia sobre el cuidado del medio ambiente y las implicaciones del uso irracional de los combustibles, a través de las actividades cotidianas.*

¿Qué debemos hacer?

Considera la sintaxis y las estructuras de control estudiadas en las lecturas 3. Instrucciones en Java para aplicaciones básicas, 4. Estructuras de Java, 5. Clases y objetos. Deberás realizar la investigación correspondiente a la **huella de carbono**, los gases de efecto invernadero y el cálculo en términos de dióxido de carbono equivalente (eCO_2e) Para después utilizar el IDE de tu preferencia como software para desarrollar su propuesta de programa, recuerda tener presente las recomendaciones para reducir la cantidad de huella generada en el hogar o por la persona.

Al final, realicen y socialicen la reflexión a través de las preguntas del conflicto cognitivo.

1. ¿De qué manera se pueden relacionar el desarrollo de software con los modelos matemáticos y los fenómenos del cambio climático?
2. ¿Qué se requiere para generar una propuesta de software para calcular la huella de CO2 de una familia o persona?
3. ¿Qué recomendaciones darías para la reducción de emisiones y la eficiencia en el uso de recursos?
4. ¿Cuáles son las diferencias entre la programación en C++ y Java?
5. ¿Cuáles son las palabras reservadas y estructura en Java?
6. ¿Cuál es la diferencia entre clases y objetos?
7. ¿En qué dispositivos se ha utilizado Java?

ANEXOS

HIMNO DEL COBATAB

¡Oh!, Colegio de Bachilleres, impetuosa y querida
institución casa fiel del conocimiento,
hoy te canto este himno con amor,
eres rayo de esperanza del mañana,
eres la voz de la verdad.

¡Oh!, Colegio de Bachilleres, eres
luz en medio de la oscuridad.
//Colegio de bachilleres conducta
clara y firme decisión
Colegio de bachilleres tu misión
para siempre es ser mejor.//

En Tabasco se ha sembrado
la semilla que un día germinara,
el impulso de la vida modernista
en progreso de toda la sociedad,
es tu memorable historia gran
orgullo para toda la región,
educación que genera cambio,
ejemplo digno en cada generación.

PORRA INSTITUCIONAL

¡Somos!
¡Somos!

Jóvenes Bachilleres
Jóvenes Bachilleres

Con Valor y Lealtad
De Norte a Sur
De Este a Oeste

Somos líderes Bachilleres del Sureste
COBATAB Unido, COBATAB Fortalecido.
Este encuentro lo gano porque lo gano
Como dijo el Peje, me canso ganso.

¡Somos!
¡Somos!

Jóvenes Bachilleres
Jóvenes Bachilleres

¡Somos!
¡Somos!

Jóvenes Bachilleres
Jóvenes Bachilleres

COBATAB Unido, COBATAB Fortalecido

COBACHITO

Colegio de Bachilleres,
Está de fiesta señores
Pues todos sus estudiantes
Hoy celebran con honores.

Que ya llegó la alegría
Es hora de motivar
Bailemos con algarabía
Cobachito nos guiará.

Allá por el acahual
En los ríos de Tabasco
Aconchado en unas ramas
O nadando sin parar.

Un manatí se ha ganado
El cariño de la gente
Cobachito le han llamado
Y no para de bailar.

Cobachito, con él vamos a ganar
Cobachito, eres espectacular
Cobachito, respetamos tu hábitat
Cobachito, mascota del COBATAB.

Mientras la orquesta se escucha
Y la porra se emociona
Los jóvenes bachilleres
A una voz ovacionan.

Con orgullo representan
A una gran institución
COBATAB está presente
Y Cobachito ya llegó.

COBATAB

COLEGIO DE BACHILLERES
DE TABASCO