UNIVERSIDAD DE GUADALAJARA

1. DATOS GENERALES DE LA UNIDAD DE APRENDIZAJE (UA) O ASIGNATURA			
Nombre de la Unidad de Aprendizaje (UA) o Asignatura			**Clave de la UA**
Programación para Ciencias			I5937
Modalidad de la UA	**Tipo de UA**	**Área de formación**	**Valor en créditos**
Escolarizada	Taller	Básica común obligatoria	5

UA de pre-requisito	**UA simultaneo**	**UA posteriores**
Ninguno	Ninguno	IB057 Métodos Numérico
Horas totales de teoría	**Horas totales de práctica**	**Horas totales del curso**
0	68	68

Licenciatura(s) en que se imparte	**Módulo al que pertenece**
Licenciatura en Matemáticas	Métodos Numéricos
Departamento	**Academia a la que pertenece**
Departamento de Matemáticas	Modelación Matemática
Elaboró	**Fecha de elaboración o revisión**
José Francisco Villalpando Becerra	07/07/2017

2. DESCRIPCIÓN DE LA UA O ASIGNATURA

Presentación

La Universidad de Guadalajara, consciente de la necesidad de vincular el aprendizaje de sus estudiantes con las actividades laborales, ha emprendido una reforma curricular, en la que se enfatiza el desarrollo de habilidades cognitivas de orden superior (pensamiento analítico, pensamiento crítico, solución de problemas y comunicación), habilidades de pensamiento complejo, alfabetización Informacional, capacidad para organizar, gestionar el tiempo, tomar decisiones y trabajar colaborativamente, responsabilidad social y creatividad.

Entre los objetivos propuestos se encuentran: potenciar y desarrollar la competencia matemática, entendiendo como competencia matemática el estudiar, analizar y reproducir resultados y nuevas teoría para establecer los límites de la matemática actual en una determinada sub-disciplina. Establecer relaciones entre distintos puntos de vista o enfoques de un mismo tópico matemático. Comunicar ideas y teorías matemáticas con otros expertos en matemáticas.

Por lo anterior se considera que un matemático, entre muchas cosas más, debe ser capaz de elaborar Cómputo Científico, al utilizar la computadora como una herramienta auxiliar en el análisis de problemas y diseño de soluciones. Además de analizar y validar los resultados obtenidos por una computadora. Así como análisis y diseño de algoritmos computacionales (simbólicos y numéricos). Es decir, un matemático debe hacer matemáticas con la computadora.

Relación con el perfil

Modular	De egreso
La materia de Programación para Ciencias fue creada para introducir al estudiante al mundo del cómputo científico apoyado con el uso de software para proponer modelos matemáticos y computacionales aplicables en la matemática. Aplica al perfil del módulo Métodos Numéricos de la Licenciatura en Matemáticas.	La materia de Programación para Ciencias abona al fortalecimiento, en la Licenciatura en Matemáticas, en la competencia "Usa herramientas de cómputo científico, entendiendo los algoritmos utilizados y las particularidades de los resultados obtenidos" del perfil de egreso.

Competencias a desarrollar en la UA o Asignatura

Transversales	Genéricas	Profesionales
Utiliza el lenguaje formal de la Matemática para la solución de problemas mediante el uso de software libre para la programación. Resuelve problemas mediante programas, de manera autónoma y colaborativamente en base a la complejidad de los mismos utilizando software libre.	Utiliza software libre para modelar matemáticamente la solución de problemas. Involucra el uso de la computadora como una herramienta diaria de trabajo.	Colabora con otros profesionales para resolver problemas reales utilizando software libre. Utiliza las Tecnologías de la Información y Comunicación en la solución de problemas. Transfiere los conocimientos adquiridos con el uso del software libre a la solución de problemas.

Saberes involucrados en la UA o Asignatura

Saber (conocimientos)	Saber hacer (habilidades)	Saber ser (actitudes y valores)
Definición común y formal e algoritmo, complejidad de algoritmos, a análisis asintótico de funciones, análisis de algoritmos. Diseño de programas y fases para el diseño de programas. Fases para la puesta a punto de programas. Programación estructurada, estructuras básicas de	Organiza los datos requeridos para la solución de un problema. Emplea adecuadamente las herramientas matemáticas computacionales dependiendo del área de la misma a la que se refiera el problema en cuestión.	Entrega en tiempo y forma los resultados de las actividades propuestas para el curso. Muestra interés y honestidad al realizar las actividades del curso. Acata los acuerdos tomados por el grupo

control. Objetos de un programa, expresiones, constantes, variables, orden de evaluación de los operadores. Estructura general de un programa, partes de un programa, diferentes tipos de instrucciones, elementos auxiliares de un programa. Fundamentos de programación en Fortran, tipos de datos, constantes, variables, operaciones en Fortran, Arreglos, funciones intrínsecas, control de flujo del programa. Programación modular en Fortran, Funciones, subrutinas, módulos. Compilación por separado de las unidades del programa. Entrada y salida de archivos en Fortran. Fundamentos de programación en Octave, condiciones y ciclos, sentencias If, Switch, For, Repeat-Until, While y Break. Archivos .m en Octave. Definición de funciones en Octave.	Justifica la elaboración de un programa cuando el caso lo requiera. Redacta respetando reglas ortográficas. Traduce a nivel básico de inglés.	o cuando así sea requerido. Respeta las ideas de sus compañeros cuando no concuerden con la propia. Entrega las actividades con claridad y limpieza.

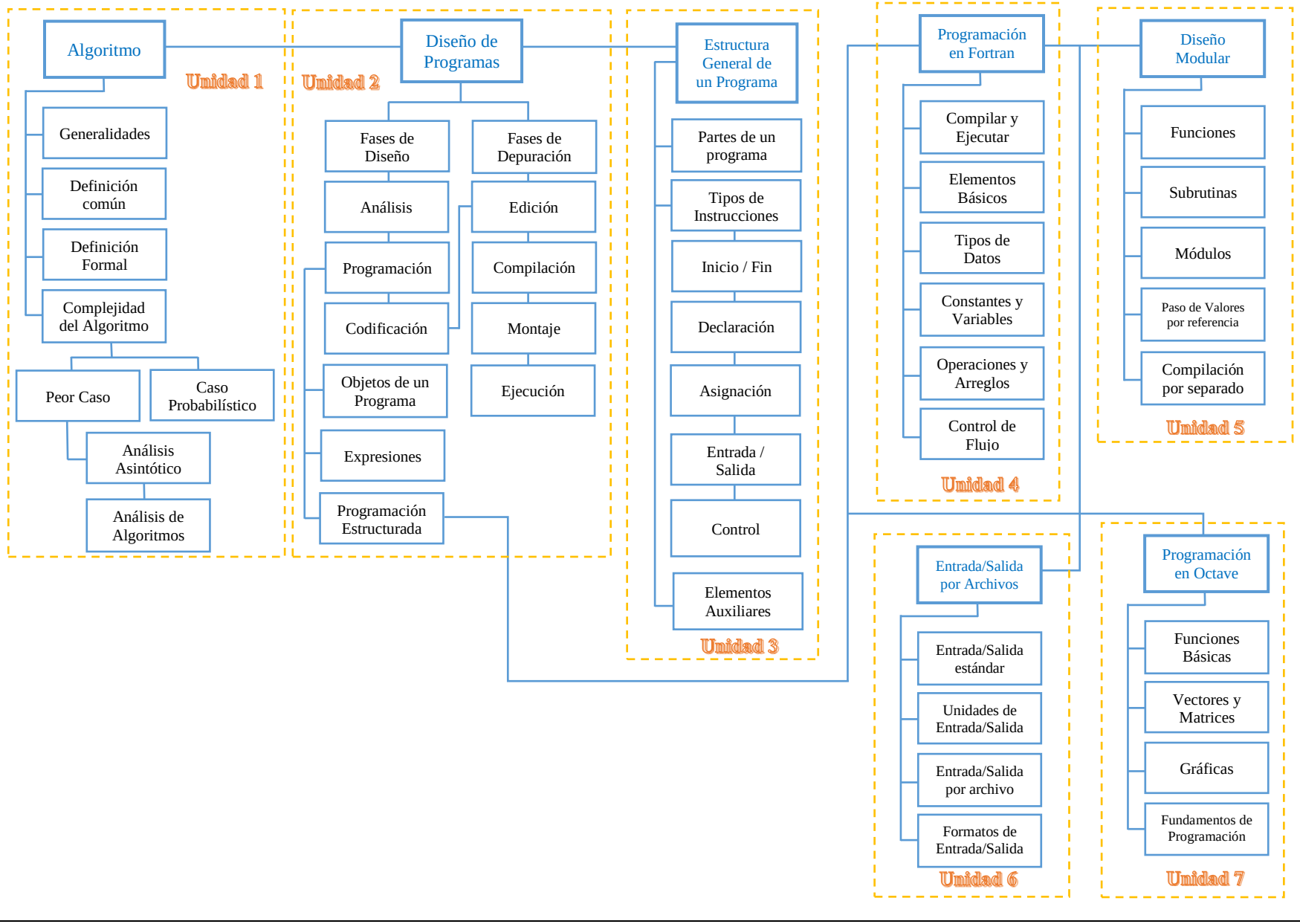
Producto Integrador Final de la UA o Asignatura

Título del Producto: Programación de algoritmos en Fortran

Objetivo: Utilizar los conocimientos adquiridos de la técnica de programación denominada Programación Estructurada y aplicarlos en la realización de programas en Fortran.

Descripción: Programar en Fortran los 9 algoritmos que vienen al final del Manual de Programación para Ciencias, siguiendo la técnica de programación conocida como Programación Estructurada, de manera tal que estén todos incluidos en un menú y el usuario decida cuál utilizar, usándolo las veces que sea necesario. La ejecución del programa se terminará cuando se dé la opción “salir”. Para esto hacer un programa principal en un archivo y todos los algoritmos en otro archivo, utilizando para ello funciones o subrutinas, para finalmente compilar por separado cada unidad.

3. ORGANIZADOR GRÁFICO DE LOS CONTENIDOS DE LA UA O ASIGNATURA



4. SECUENCIA DEL CURSO POR UNIDADES TEMÁTICAS
Unidad temática 1: Introducción a la Algoritmia (5 hrs)
Objetivo de la unidad temática: Aplicar los elementos básicos de la algoritmia en el análisis de diversos algoritmos de complejidad moderada. **Introducción:** Conforme los computadores se vuelven más y más rápidos, se torna más aguda la necesidad de algoritmos que puedan manejar grandes cantidades de datos de entrada. Paradójicamente esto requiere una mayor atención en la eficiencia, ya que la ineficiencia en los algoritmos es más obvia cuando es grande el tamaño de la entrada.

Para que un computador lleve a cabo una tarea o la resolución de un problema específico es necesario indicarle qué operaciones debe realizar, en otras palabras, debemos describir cómo debe resolver el problema. Tal descripción se llama "algoritmo". Pero, dicho algoritmo qué tan eficiente será. Existen muchos métodos y criterios para comprobar la eficiencia de un algoritmo. El más utilizado, por la mayoría de los expertos, consiste en determinar o medir el tiempo de ejecución del mismo. El Análisis de Algoritmos será de gran utilidad para conocerlo, ya que una de sus funciones principales es de la determinar la eficiencia de un algoritmo, basándose principalmente entre otras cosas, en el tiempo de ejecución que tarda el mismo para llevar a cabo su tarea.

El tiempo real requerido por un computador para ejecutar un algoritmo es directamente proporcional al número de operaciones "básicas" que el computador debe realizar en su ejecución. Por lo tanto, medir el tiempo real equivale a medir el número de operaciones elementales realizadas, además el tiempo de ejecución también equivale al tiempo utilizado por el computador. Si el tiempo de ejecución es bastante grande, puede suceder que el algoritmo en la práctica resulte inútil, pues el tiempo necesario para su ejecución puede sobrepasar el tiempo disponible para el uso del computador.

El tiempo de ejecución depende no sólo del algoritmo base, sino también del conjunto de instrucciones elaboradas por el programador. El programa también puede adaptarse para trabajar correctamente sobre un conjunto particular de datos de entrada de prueba. Estas dependencias suelen ser muy notorias con un computador, un compilador, un programador o un conjunto de entradas de datos de prueba distintos.

Para superar esos inconvenientes, los expertos en computación han adoptado la complejidad de tiempo asintótico, como la medida fundamental de la eficiencia de un algoritmo.

El término eficiencia se referirá a esta medida y en especial a la complejidad del tiempo (también conocida como orden asintótico) en el peor caso T(n), en contraposición al promedio A(n). El método para determinar la complejidad el tiempo asintótico se conoce como Análisis Asintótico.

Contenido temático	Saberes involucrados	Producto de la unidad temática
1.1 Introducción 1.2 Algoritmos: generalidades 1.3 Definición común de algoritmo 1.3.1 Ejemplo de un algoritmo conocido 1.4 Definición formal de algoritmo 1.5 Complejidad de un algoritmo: Tiempo y Espacio 1.5.1 Peor caso y caso probabilístico 1.6 Análisis asintótico de funciones 1.6.1 Relaciones de dominación 1.6.2 Notaciones asintóticas 1.6.3 Funciones de complejidad de tiempo más usuales 1.7 Análisis de diversos algoritmos 1.7.1 Reglas generales para el cálculo de T(*n*) 1.7.2 ¿Cómo obtener T(*n*)? 1.7.3 Análisis asintótico de las funciones obtenidas 1.7.4 Completando el análisis asintótico	Conceptualizar el concepto de Algoritmo intuitivamente. Formalizar el concepto de algoritmo. Conocer diferencia entre los conceptos de tiempo y espacio en la complejidad computacional. Conocer los elementos básicos del análisis asintótico de funciones. Aplicar los elementos básicos de la Algoritmia y del análisis asintótico de funciones en el análisis de algoritmos de complejidad moderada.	Materiales para el curso: programas Fortran y Octave y algunos manuales, así como el manual de la Materia de Programación para Ciencias. Equipos formados para las exposiciones de las unidades 2 y 3. Resolución de las actividades de la unidad 1 del Manual de Programación para Ciencias, consistente en la obtención de la complejidad computacional de los algoritmos: a) máximo versión 2 b) búsqueda binaria c) mezclas Los cuales están explicados en el manual de la materia.

Unidad temática 2: Diseño de programas (4 hrs)

Objetivo de la unidad temática: Diferenciar las fases en el diseño y depuración de un programa.

Introducción: En la vida cotidiana nos enfrentamos a problemas que debemos resolver en lo posible eficazmente. Así como cada individuo tiene formas

de encarar un problema y su propia manera de solucionarlo, computacionalmente hablando se puede hacer una analogía.
Ante la presentación de un problema encarar la mejor forma de resolverlo para arribar al resultado esperado y correcto es un desafío. Para ello debemos comprender exactamente qué se pide, qué resultados se pretenden y que restricciones y/o condiciones existen. Para realizar lo antes dicho se divide la resolución de un problema por medio de un programa en etapas.

Contenido temático	Saberes involucrados	Producto de la unidad temática
2.1 Introducción 2.2 Fases para el diseño de un programa 2.2.1 Análisis 2.2.2 Programación 2.2.3 Codificación 2.3 Herramientas de programación 2.3.1 Reglas para la creación de diagramas de flujo 2.4 Ejemplo del diseño de un programa en Fortran 2.5 Fases para la puesta a punto del programa (depuración) 2.5.1 Edición 2.5.2 Compilación 2.5.3 Montaje 2.5.4 Ejecución 2.6 Características de un programa 2.6.1 Características generales de un programa 2.7 Programación estructurada 2.7.1 Definición de las estructuras básicas 2.8 Objetos de un programa: constantes y variables 2.8.1 Atributos de un objeto 2.8.2 Constantes 2.8.3 Variables 2.9 Expresiones 2.9.1 Tipos de expresiones 2.9.2 Operadores 2.9.3 Tabla de verdad de los operadores lógicos 2.9.4 Orden de evaluación de los operadores 2.10 Tipo de errores de cálculo 2.10.1 Cifras significativas 2.10.2 Exactitud y precisión 2.10.3 Error 2.10.4 Error de redondeo 2.10.5 Error de truncamiento 2.10.6 Error numérico total	Conocer y diferenciar las fases en el diseño de un programa. Conocer algunas herramientas de programación. Conocer y diferenciar las fases en la depuración de un programa. Conocer las características de un programa. Conocer los objetos básicos de un programa. Evaluar diversas expresiones de acuerdo a su orden de evaluación de los operadores. Diferenciar los tipos de errores de cálculo al ejecutar un programa. Elaborar programas de complejidad moderada, sin la fase de codificación y edición.	Resolución de las actividades de la unidad 2 del Manual de Programación para Ciencias, consistente en la elaboración de 4 programas de complejidad moderada, sin la fase de codificación y edición. Dichos programas son: a) Leer dos números y calcular e imprimir su suma, resta, producto y división. b) Dado un número que corresponda al radio de una circunferencia y calcular e imprimir la longitud de la misma y el área del círculo correspondiente. c) Obtener una ecuación de primer grado: Y = A * X + B. En primer lugar se leen los coeficientes A y B y, a continuación, se leen dos valores de la variable X, obteniendo como resultado las coordenadas cartesianas X, Y de dos puntos, que nos permitan representar la recta correspondiente a la ecuación. d) Obtener las raíces de un polinomio de segundo grado utilizando la fórmula general.

Unidad temática 3: Estructura general de un programa (3 hrs)

Objetivo de la unidad temática: Identificar los elementos involucrados en la estructura general de un programa.

Introducción: Un programa puede considerarse como una secuencia de acciones o instrucciones que manipulan un conjunto de objetos o datos. En otras palabras un programa es un algoritmo llevado a la práctica en un lenguaje de programación.
Tras la decisión de desarrollar un programa, el programador debe establecer el conjunto de especificaciones que debe contener el programa: entradas, salidas, algoritmos de resolución, que incluirán las técnicas para obtener las salidas a partir de las entradas.
Todo programa deberá contener dos bloques para la descripción de los aspectos descritos anteriormente.

Contenido temático	Saberes involucrados	Producto de la unidad temática
3.1 Introducción 3.2 Partes principales de un programa 3.3 Clasificación de instrucciones 3.3.1 Instrucciones de inicio/fin 3.3.2 Instrucciones de declaración 3.3.3 Instrucciones de asignación 3.3.4 Instrucción de entrada 3.3.5 Instrucción de salida 3.3.6 Instrucciones de control 3.4 Elementos auxiliares de un programa	Conocer y diferenciar las partes principales de un programa. Conocer y diferenciar diversos tipos de instrucciones de un programa. Conocer y diferenciar los elementos auxiliares de un programa	Al final de la unidad del alumno realizará una búsqueda por equipos para exponer, sobre los diversos tipos de programas que existen y su clasificación, además de los lenguajes de programación en que pueden realizarse los mismos, sabiendo que, en general, los programas se clasifican en dos tipos: 1) Programas de sistemas y 2) Programas de aplicación

Unidad temática 4: Fundamentos de programación en Fortran (27 hrs)

Objetivo de la unidad temática: Diseñar y codificar algoritmos de complejidad moderada en el lenguaje de programación Fortran.

Introducción: Fortran es el lenguaje de programación de más amplio uso en el cómputo científico. El nombre FORTRAN proviene de "FORmula TRANslator" (traductor de fórmulas), y fue desarrollado originalmente por IBM en 1954, con el objetivo de poder escribir programas de cómputo científico en un lenguaje de alto nivel en vez de tener que recurrir a lenguaje de máquina o ensamblador. En 1958 se presentó una segunda versión y varias compañías comenzaron a desarrollar compiladores independientes a IBM para usar el lenguaje en otras máquinas.
El primer estándar de Fortran se introdujo en 1962 y se llamó Fortran IV. En 1966 se presentó el primer estándar ANSI (American National Standards Institute), que se conoció como Fortran 66. El segundo estándar ANSI, con muchas mejoras, se introdujo en 1977 (Fortran 77), y se convirtió en el estándar utilizado por la comunidad científica por muchos años. Incluso a la fecha es común encontrar muchos programas escritos en Fortran 77.
Fortran 77 tenía una serie de desventajas. Entre ellas una estructura muy rígida adaptada al uso de tarjetas perforadas ("forma fija"), que requería que ciertas columnas tuvieran usos específicos. Además, no permitía un uso dinámico de la memoria y no permitía realizar operaciones entre arreglos de números. Para mejorar esta situación, en 1990 se presentó un tercer estándar ANSI conocido como Fortran 90, que contenía muchas nuevas características y permitía una programación más estructurada. Una serie de cambios menores se presentaron en 1995 (Fortran 95), y actualmente se trabaja en un nuevo estándar ANSI (Fortran 2003).
Actualmente la mayor parte de los programas en Fortran siguen el estándar de Fortran 90, pero aún existe un número importante de aplicaciones de Fortran 77. Fortran esta específicamente diseñado para el cómputo científico, y no es particularmente bueno para otro tipo de aplicaciones (control, administración, manejo de documentos, etc.). Para estas aplicaciones otros lenguajes como C, JAVA o PERL son más adecuados. En la actualidad, la mayor parte del cómputo científico de alto rendimiento a nivel internacional se lleva a cabo en Fortran (Fortran está muy lejos de ser obsoleto), aunque los lenguajes C y

C++ han ganado cierta popularidad recientemente.

Contenido temático	Saberes involucrados	Producto de la unidad temática
4.1 Breve historia de Fortran 4.2 Compilar y ejecutar programas en Fortran 4.3 Elementos básicos de un programa en Fortran 4.4 Tipos de Datos 4.5 Constantes 4.5.1 Constantes enteras 4.5.2 Constantes reales 4.5.3 Constantes de doble precisión 4.5.4 Constantes lógicas 4.5.5 Constantes complejas 4.5.6 Constantes carácter 4.5.7 Constantes con nombre: PARAMETER 4.6 Declaración de variables 4.6.1 Conversión entre tipos 4.7 Operaciones en Fortran 4.8 Arreglos 4.8.1 Arreglos de tamaño fijo 4.8.2 Asignación dinámica de memoria 4.9 Funciones intrínsecas 4.10 Control de flujo del programa 4.10.1. Ciclos 4.10.2 IF 4.10.3 SELECT CASE 4.10.4 Control lógico de ciclos 4.10.5 EXIT y CYCLE	Conocer los fundamentos de programación en Fortran Diseñar y codificar algoritmos de complejidad moderada para que puedan ser codificados en Fortran. Compilar y ejecutar programas en Fortran. Corregir los errores que se pueden presentar al codificar y/o ejecutar los programas en Fortran.	A los largo del tiempo asignado para la unidad, el alumno realizará las 19 actividades diseñadas para ser programadas en fortran, las cuales vienen al final de la unidad 4 del Manual de Programación para Ciencias. No se admitirá ninguna actividad fuera de la fecha señalada. Los archivos de la actividad deberán seguir la nomenclatura act04_yy.f90 donde yy el número del ejercicio de la actividad. Adicionalmente cada archivo deberá contener internamente el nombre, código y sección del alumno. Todos los ejercicios de cada actividad deberán ser compactados en un sólo archivo en formato .zip o .rar y enviados por e-mail a la dirección indicada por el profesor. En el asunto del e-mail se deberá incluir la palabra Programación, nombre del alumno y sección.

Unidad temática 5: Diseño descendente de programas (12 hrs)

Objetivo de la unidad temática: Utilizar los elementos básicos de la programación estructurada en el diseño de programas en Fortran.

Introducción: Frente a un problema complejo una de las maneras más eficientes de diseñar (e implementar) un algoritmo para el mismo consiste en descomponer dicho problema en subproblemas de menor dificultad y éstos, a su vez, en subproblemas más pequeños y así sucesivamente hasta cierto grado de refinamiento donde cada subproblema involucre una sola tarea específica bien definida y, preferiblemente, independiente de los otros. El problema original es resuelto, entonces, combinando apropiadamente las soluciones de los subproblemas.

En una implementación computacional, cada subproblema es implementado como un subprograma. De este modo el programa consta de un programa principal (la unidad del programa de nivel más alto) que llama a subprogramas (unidades del programa de nivel más bajo) que a su vez pueden llamar a otros subprogramas.

En muchas ocasiones existen tareas que se deben realizar muchas veces durante la ejecución de un código, por lo que resulta conveniente que sean subprogramas en sí mismos. La existencia de subprogramas también hace que un código sea más modular, y permite que diferentes personas programes

diferentes partes de un mismo código.
Los diversos subprogramas pueden ser parte de un mismo archivo, o estar en archivos separados que se unen durante el proceso de compilación. Si están en un mismo archivo, deben aparecer después de la línea que termina el programa principal. En FORTRAN 90 hay tres tipos básicos de subprogramas: funciones, subrutinas y módulos.

Contenido temático	Saberes involucrados	Producto de la unidad temática
5.1 Programación descendente o modular en Fortran 5.2 Implementación de subprogramas en Fortran 5.2.1 Funciones 5.2.2 Subrutinas 5.2.3 Módulos 5.3 Características de las subrutinas y funciones 5.4 Paso de valores por referencia 5.5 Compilación por separado de las unidades del programa	Codificar y ejecutar programas en Fortran utilizando el diseño descendente de programas para la modularidad del mismo. Compilar por separado las unidades (funciones, subrutinas y módulos) para obtener un programa modular, siguiendo los lineamientos de la programación estructurada. Corregir los errores que se pueden presentar al codificar y/o ejecutar los programas en Fortran.	A los largo del tiempo asignado para la unidad, el alumno realizará las 8 actividades diseñadas para ser programadas en fortran, las cuales vienen al final de la unidad 5 del Manual de Programación para Ciencias. No se admitirá ninguna actividad fuera de la fecha señalada. Los archivos de la actividad deberán seguir la nomenclatura act05_yy.f90 donde yy el número del ejercicio de la actividad. Adicionalmente cada archivo deberá contener internamente el nombre, código y sección del alumno. Todos los ejercicios de cada actividad deberán ser compactados en un sólo archivo en formato .zip o .rar y enviados por e-mail a la dirección indicada por el profesor. En el asunto del e-mail se deberá incluir la palabra Programación, nombre del alumno y sección.

Unidad temática 6: Entrada y Salida por Archivos en Fortran (6 hrs)

Objetivo de la unidad temática: Elaborar programas en Fortran que involucren la utilización de archivos.

Introducción: Para problemas que involucren grandes cantidades de datos resulta más conveniente que los mismos estén guardados en archivos. Un archivo es un conjunto de datos almacenado en un dispositivo (tal como un disco duro). Para la mayoría de las aplicaciones los únicos tipos de archivos que interesa considerar son los archivos de texto. Un archivo de texto consta de una serie de líneas o registros separadas por una marca de fin de línea (newline, en inglés).
Cada línea consta de uno o más datos que es un conjunto de caracteres alfanuméricos que, en el procesamiento de lectura o escritura, se trata como una sola unidad. El acceso a los datos del archivo de texto procede en forma secuencial, esto es, se procesan línea por línea comenzando desde la primera línea hacia la última. Esto implica que no es posible acceder a una línea específica sin haber pasado por las anteriores.

Contenido temático	Saberes involucrados	Producto de la unidad temática
6.1 Entrada y salida de datos estándar 6.2 Unidades de entrada y salida 6.3. Entrada y salida de datos por archivo 6.4. Formato de entrada y salida	Conocer las unidades lógicas de entrada y salida para archivos en Fortran. Codificar y ejecutar programas en Fortran que requieran la utilización de archivos. Corregir los errores que se pueden presentar al codificar y/o ejecutar los programas en Fortran.	A los largo del tiempo asignado para la unidad, el alumno realizará las 2 actividades diseñadas para ser programadas en fortran, las cuales vienen al final de la unidad 6 del Manual de Programación para Ciencias. No se admitirá ninguna actividad fuera de la fecha señalada. Los archivos de la actividad deberán seguir la nomenclatura act06_yy.f90 donde yy el número del ejercicio de la actividad. Adicionalmente cada archivo deberá contener internamente el nombre, código y sección del alumno. Todos los ejercicios de cada actividad deberán ser compactados en un sólo archivo en formato .zip o .rar y enviados por e-mail a la dirección indicada por el profesor. En el asunto del e-mail se deberá incluir la palabra Programación, nombre del alumno y sección.

Unidad temática 7: Fundamentos de programación en Octave (12 hrs)

Objetivo de la unidad temática: Diseñar, codificar y ejecutar algoritmos de complejidad moderada en el software libre para cálculo numérico Octave

Introducción: Octave o GNU Octave es un programa libre para realizar cálculos numéricos. Como indica su nombre es parte de proyecto GNU. MATLAB® es considerado su equivalente comercial. Entre varias características que comparten se puede destacar que ambos ofrecen un intérprete permitiendo ejecutar órdenes en modo interactivo. Octave no es un sistema de álgebra computacional como podría ser Maxima, sino que usa un lenguaje que está orientado al análisis numérico.

El proyecto fue creado alrededor del año 1988 pero con una finalidad diferente: ser utilizado en un curso de diseño de reactores químicos en la Universidad de Texas. Posteriormente en el año 1992, se decide extenderlo y comienza su desarrollo a cargo de John W. Eaton. La primera versión alpha fue lanzada el 4 de enero de 1993. Un año más tarde, el 17 de febrero de 1994 aparece la versión 1.0.

El nombre surge de Octave Levenspiel, profesor de algunos de los autores conocido por sus buenas aproximaciones por medio de cálculos mentales a problemas numéricos en ingeniería química.

La flexibilidad de este programa en seguida lo hizo popular y su uso se expandió a otros problemas relacionados con el álgebra lineal y las ecuaciones diferenciales y favoreció su desarrollo, agregando las aportaciones de la comunidad de usuarios.

Contenido temático	Saberes involucrados	Producto de la unidad temática
7.1 Breve historia de Octave 7.2 Funciones básicas de Octave 7.2.1 Operadores aritméticos y funciones matemáticas elementales 7.2.2 Funciones matemáticas elementales 7.2.3 Operadores de comparación 7.2.4 Operadores booleanos 7.2.5 Operadores booleanos "short-circuit" 7.2.6 Operador de asignación 7.3 Vectores y matrices 7.3.1 Matrices especiales 7.3.2 Operaciones con matrices y vectores 7.4 Gráficas 7.4.1 Gráficas en dos dimensiones 7.4.2 Gráficas tridimensionales 7.4.3 Múltiples gráficas 7.5. Programación en Octave 7.5.1 Condicionales y ciclos 7.5.2 Sentencia IF 7.5.3 Sentencia SWITCH 7.5.4 Sentencia FOR 7.5.5 Sentencia DO-UNTIL 7.5.6 Sentencia WHILE 7.5.7 Sentencia BREAK y CONTINUE 7.6 Archivos *.m 7.6.1 Archivos de Comandos (SCRIPTS) 7.6.2 Definición de Funciones 7.6.3 HELP para las funciones de usuario	Conocer la historia del software libre para cálculo numérico Octave. Conocer cómo se instala el programa Octave. Conocer las funciones básicas de Octave. Conocer los fundamentos de programación en Octave. Diseñar y codificar algoritmos de complejidad moderada para que puedan ser codificados en Octave. Ejecutar programas en Octave. Corregir los errores que se pueden presentar al codificar y/o ejecutar un programa en Octave.	A los largo del tiempo asignado para la unidad, el alumno realizará las 5 actividades diseñadas para ser programadas en Octave, las cuales vienen al final de la unidad 7 del Manual de Programación para Ciencias. No se admitirá ninguna actividad fuera de la fecha señalada. Los archivos de la actividad deberán seguir la nomenclatura act07_yy.m donde yy el número del ejercicio de la actividad. Adicionalmente cada archivo deberá contener internamente el nombre, código y sección del alumno. Todos los ejercicios de cada actividad deberán ser compactados en un sólo archivo en formato .zip o .rar y enviados por e-mail a la dirección indicada por el profesor. En el asunto del e-mail se deberá incluir la palabra Programación, nombre del alumno y sección.

5. EVALUACIÓN Y CALIFICACIÓN

Requerimientos de acreditación:

Para que el alumno tenga derecho al registro del resultado final de la evaluación en el periodo ordinario el alumno debe tener un mínimo 80% tanto de asistencia a clases como de actividades registradas durante el curso. Para aprobar la Unidad de Aprendizaje el estudiante requiere una calificación mínima de 60.

Criterios generales de evaluación:

La entrega de cada actividad deberá en tiempo indicado.
Las actividades para entregar son personales y deberán incluir una portada con los datos del curso y del alumno.
Si se detecta que una actividad fue copiada se anulará a ambos alumnos.

Evidencias o Productos

Evidencia o producto	Competencias y saberes involucrados	Contenidos temáticos	Ponderación
Actividades contestadas	Definición común y formal e algoritmo, complejidad de algoritmos, a análisis asintótico de funciones, análisis de algoritmos. Diseño de programas y fases para el diseño de programas. Fases para la puesta a punto de programas. Programación estructurada, estructuras básicas de control. Objetos de un programa, expresiones, constantes, variables, orden de evaluación de los operadores. Fundamentos de programación en Fortran, tipos de datos, constantes, variables, operaciones en Fortran, Arreglos, funciones intrínsecas, control de flujo del programa. Programación modular en Fortran, Funciones, subrutinas, módulos. Compilación por separado de las unidades del programa. Entrada y salida de archivos en Fortran. Fundamentos de programación en Octave, condiciones y ciclos, sentencias If, Switch, For, Repeat-Until, While y Break. Archivos .m en Octave. Definición de funciones en Octave.	Unidades 1, 2, 4, 5, 6 y 7	40 %
Exposición en clase	Realizar una búsqueda en internet por equipos para exponer, sobre los diversos tipos de programas que existen, además de los lenguajes de programación en que pueden realizarse los mismos, sabiendo que, en general, los programas se clasifican en dos tipos: 1) Programas de sistemas y 2) Programas de aplicación	Unidad 3	10 %

Producto final

<table>
<tr><th colspan="3">Descripción</th><th colspan="2">Evaluación</th></tr>
<tr><td colspan="3">Título: Programación de algoritmos en Fortran</td><td rowspan="3">Criterios de fondo:
Programar los 9 algoritmos en Fortran utilizando la técnica de programación denominada Programación Estructurada
Criterios de forma:
Se realizará un archivo .f90 el cual incluya un menú con los diferentes algoritmos a ejecutar. En otro archivo se programaran todos los algoritmos utilizando para ello funciones o subrutinas, para finalmente compilarlos por separado quedando un solo archivo ejecutable. Se enviará por e-mail al profesor los archivos .f90 y .exe resultantes.</td><td>Ponderación</td></tr>
<tr><td colspan="3">Objetivo: Utilizar los conocimientos adquiridos de la técnica de programación denominada Programación Estructurada y aplicarlos en la realización de programas en Fortran.</td><td rowspan="2">40 %</td></tr>
<tr><td colspan="3">Caracterización: Programar en Fortran los 9 algoritmos que vienen al final del Manual de Programación para Ciencias, siguiendo la técnica de programación conocida como Programación Estructurada, de manera tal que estén todos incluidos en un menú y el usuario decida cuál utilizar, usándolo las veces que sea necesario. La ejecución del programa se terminará cuando se dé la opción “salir”. Para esto hacer un programa principal en un archivo y todos los algoritmos en otro archivo, utilizando para ello funciones o subrutinas, para finalmente compilar por separado cada unidad.</td></tr>
<tr><th colspan="5">Otros criterios</th></tr>
<tr><th>Criterio</th><th colspan="3">Descripción</th><th>Ponderación</th></tr>
<tr><td>Participación en clase</td><td colspan="3">Participación activa y constante en las diferentes intervenciones</td><td>10 %</td></tr>
</table>

6. REFERENCIAS Y APOYOS				
Referencias bibliográficas				
Referencias básicas				
Autor (Apellido, Nombre)	**Año**	**Título**	**Editorial**	**Enlace o biblioteca virtual donde esté disponible (en su caso)**
Villalpando Becerra J. F.	2014	Manual de Programación para Ciencias		
Referencias complementarias				
Alcalde E. y García M.	1992	Metodología de la Programación	McGraw Hill	
Joyanes L.	2013	Fundamentos generales de Programación	McGraw Hill	
Brassard G. y Bratley P.	1997	Fundamentos de Algoritmia	Prentice Hall	
Abellanas M. y Lodares D.	1991	Análisis de Algoritmos	Macrobit /Ra-Ma	
Grassard G. y Bratley P.	2000	Fundamentos de Algoritmia	Pearson	
Apoyos (videos, presentaciones, bibliografía recomendada para el estudiante)				