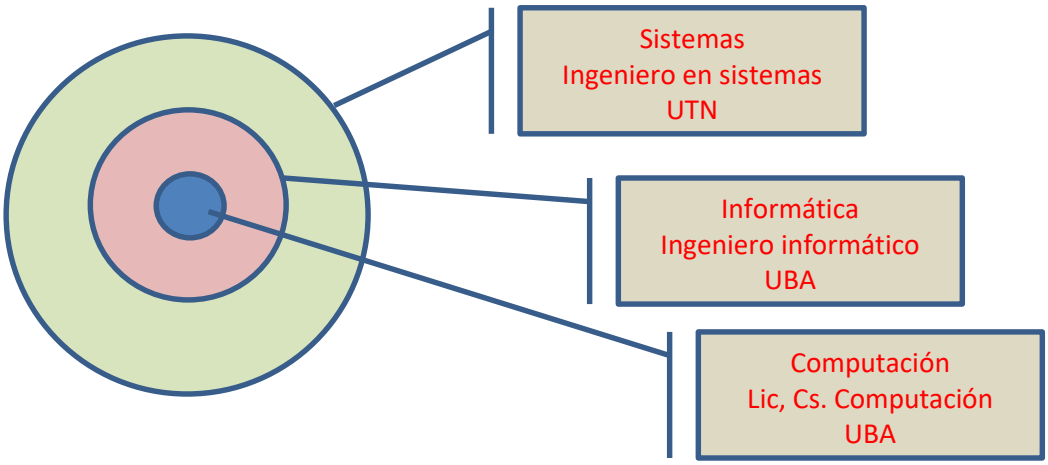
Unidad 01: Definición de términos y frases

1

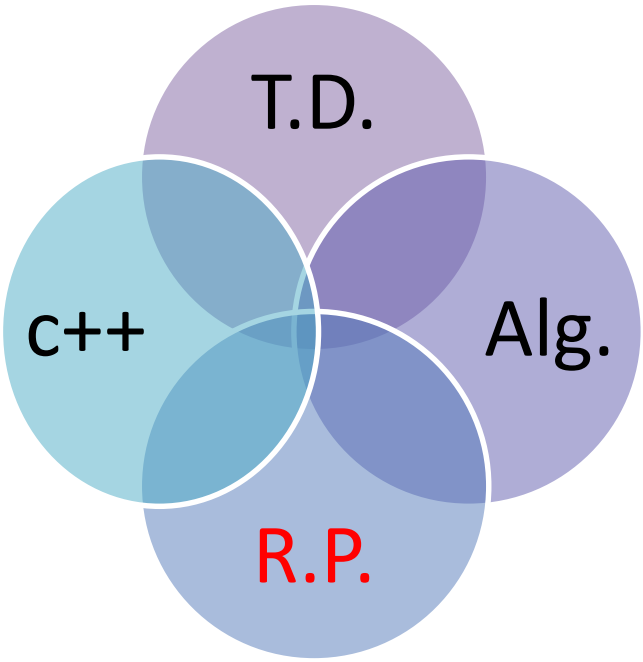
Contenidos analíticos

Parte I: Del Universo real al universo computacional

Unidad 1: Definición de términos y frases
Definición de paradigmas de programación. Profundización del paradigma imperativo, modular y estructurado. Concepto de Computación, Informática, Sistemas. Tipos de problemas: problemas computacionales. Definición de asignación, sentencia, declaración, dato, información, conocimiento, lenguajes de programación. Partes de un programa, Abstracción, Modularización, Especificación



Algoritmos y estructura de datos

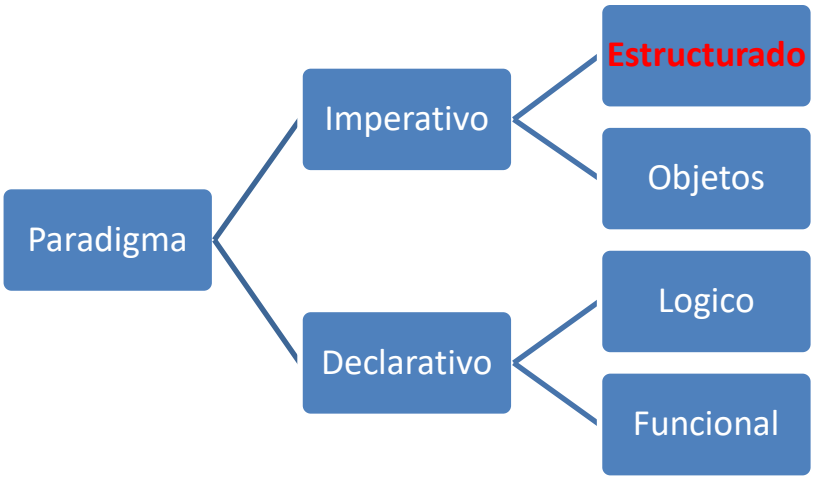


El propósito de algoritmos y estructura de datos es la resolución de problemas RP computables de información por lo que requiere la conjunción de:

1. Datos: debe definir sus tipos
2. Con una secuencia lógica y precisa de acciones: definiendo el algoritmo
3. implementar la solución: mediante en LP, en este caso C++

Paradigmas de programación

Son los diferentes modelos o criterios, incluidas las particularidades de las herramientas y métodos, para dar solución a una situación efectiva y eficiente a una situación problemática particular.



Paradigma Imperativo, modular y estructurado

Los paradigmas en programación pueden diferenciarse en dos grandes grupos: IMPERATIVO: Dar ordenes a la computadora de modo de ejecutar una serie de "sentencias" con el fin de alcanzar un objetivo o DECLARATIVO: Separar el conocimiento que se tiene sobre algo, implementando lo que se denomina "base de conocimiento "haciendo inferencias adecuadas" sobre ese conocimiento con el fin de alcanzar una solución.
El paradigma imperativo, puede a su vez ser MODULAR y ESTRUCTURADO: Dividiendo el problema en pequeños modulos (acciones), utilizando determinadas estructuras de resolución para desarrollar esas acciones y acceder a los datos, u ORIENTADO A OBJETOS en los que , con un mayor nivel de abstracción se crean objetos que contienen las acciones y las propiedades e interactúan entre ellos.

Paradigma modular y estructurado

Es modular porque el problema general se descompone en problemas menores, cuya integración alcanza la solución total todo esto con un conjunto de sentencias acotadas: Asignacion, Analisis de caso y Repeticion.

Computacion

1. Diseñar computadores; sistemas de generación, transmisión y procesamiento de señales digitales; sistemas computarizados de automatización y de control; sistemas de procesamiento y de comunicación de datos.
2. Especificar, proyectar y desarrollar software
3. Proyectar, dirigir y controlar la construcción, implementación, operación y mantenimiento
4. Certificar el funcionamiento, condición de uso o estado de los sistemas.
5. Proyectar seguridad incluyendo seguridad informática.

Informatica Sistemas

1. Especificar, proyectar y desarrollar sistemas de información y software cuya utilización pueda afectar la seguridad0, salud, bienes o derechos
2. Proyectar y dirigir lo referido a seguridad informática.
3. Establecer métricas y normas de calidad de software.
4. Certificar el funcionamiento, condición de uso o estado de lo mencionado anteriormente.
5. Dirigir y controlar la implementación, operación y mantenimiento.

Roles

1. Administrador del proyecto
    a. Project manager está a cargo del desarrollo, la organización, la entrega de un proyecto
2. Analista funcional
    a. definir y analizar procesos tecnológicos. rendimiento de sistemas existentes y desarrollar planes de cambio
3. Analista comercial
    a. Cumplir con las necesidades comerciales del cliente.
4. Arquitecto
    a. es un experto en software que toma decisiones de diseño de alto nivel

5. Diseñadores UX y UI
    a. UX (experiencia de usuario) asegura un resultado enfocado en el usuario.
    b. UI (interfaz de usuario) se enfoca en el software y en cómo lo ve el usuario. Necesitan que sea intuitivo y directo.
6. Desarrolladores de software
    a. crean el proyecto final al programar en un lenguaje de programación. Dependiendo del nivel de experiencia, pueden ser senior, medios y junior.
    b. Los desarrolladores front-end crean todo lo que el usuario final ve y con lo que interactúa.
    c. desarrolladores back-end son los responsables de los procesos y funcionalidades que están detrás de lo que el usuario ve.
    d. desarrolladores full-stack tienen un conocimiento más amplio pero menos profundo de los lenguajes de programación front-end y back-end.
7. Scrum master
    a. Verifica que el equipo siga metodologías y estructuras ágiles.
8. Testers
    a. Son los probadores de software planifican y realizan pruebas de software para comprobar si funcionan correctamente.

Programación

La programación es una actividad transversal asociada a cualquier área de la informática, aunque es la ingeniería del software el área específica que se ocupa de la creación del software.
En principio la programación se veía como un arte, solo era cuestión de dominar un lenguaje de programación, esto fue ampliamente superado

Programa:

Programa: conjunto de instrucciones no activas almacenadas en un computador, se vuelve **tarea** a partir de que se selecciona para su ejecución y permite cumplir una función específica. Un **proceso** es un programa en ejecución.

Dato

Dato representación de un objeto el mundo real mediante el cual se pueden modelizar aspectos de un problema que se desea resolver con un programa en una computadora.
<dato> -> <objeto><atributo><valor> → semáforo, luz, roja

Información

Interpretar el dato: semáforo, luz, roja → supone un alerta

Conocimiento

Que hacemos con esa información, que acciones son las apropiadas realizar en función de la información disponible y el propósito o fin a resolver, el conocimiento nos permite abordar soluciones que no necesariamente son únicas y muchas veces dependen del contexto
Semaforo, luz, roja:
→ peligro me detengo
→miro hacia ambos lados y veo de cruzar con cuidado
→en calle de doble mano miro primero a la izquierda y luego a la derecha o primero a la derecha y luego a la izquierda dependiendo del sentido de circulación

Lenguaje de programación

Conjunto de instrucciones permitidas y definidas por sus reglas sintácticas y su valor semántico para la expresión de soluciones de problemas.

Tecnologías/Frameworks FrontEnd (Web)	Lenguajes Backend
HTML (paginas web)	Javascript (*)
CSS (ESTILOS)	Python(PROPOSIVO GRAL LEGIBILIDAD INTERP)
Javascript (*)(AGREGA INTERACCION)	PHP(INDEPENDIENTE DE LA PLATAFORMA)
React(BIBLIOTECA JS)	Ruby(GRAL OO INTER.)
Redux(BIBLIOTECA INTERFACE USUARIO)	C C++ C# JavaVB
Angular(CREAR Y MANTENER PAGINAS WEB)	Go(MULTIPLAT. NO 100% OBJETO NO TD GENERICO)
Bootstrap(aplicaciones web)	

Problema: Enunciado con una incógnita

Tipo de problema: no computables; **Computables** (tratables, intratables)

Etapas de resolución de problemas con computadoras.

1. Análisis del problema: en su contexto del mundo real.
2. Diseño de la solución: Lo primero es la modularización del problema, es decir la descomposición en partes con funciones bien definidas y datos propios estableciendo la comunicación entre los módulos.
3. Especificación del algoritmo: La elección adecuada del algoritmo para la función de cada modulo es vital para la eficiencia posterior.
4. Escritura del programa: Un algoritmo es una especificación simbólica que debe convertirse en un programa real sobre un lenguaje de programación concreto.
5. Verificación: una vez escrito el programa en un lenguaje real y depurado los errores sintácticos se debe verificar que su ejecución conduzca al resultado deseado con datos representativos del problema real.

Algoritmo

Algoritmo
Especificación rigurosa (debe expresarse en forma univoca) de la secuencia de pasos, instrucciones, a realizar sobre un autómata para alcanzar un resultado deseado en un tiempo finito. Esto último supone que el algoritmo empieza y termina, en el caso de los que no son de tiempo finito (ej. Sistemas en tiempo real) deben ser de número finito de instrucciones.

Un algoritmo debe tener al menos las siguientes características:

1. **Ser preciso**:
2. **Ser definido**.
3. **Ser finito**:
4. **Presentación formal**:
5. **Corrección**:
6. **Eficiencia**:

Propiedades de los algoritmos

1. Especificación precisa de la entrada:
2. Especificación precisa de cada instrucción:
3. Un algoritmo debe ser exacto y correcto, tener etapas bien definidas y concretas.
4. Debe ser fácil de entender, codificar y depurar.
5. Debe hacer uso eficiente de los recursos de la computadora

Identificadores

Nombre simbolico que define quien programa para denotar "identificar" ciertos elementos en la aplicación o programa.
Estos elementos pueden ser:

1. Variables →asociado a un espacio de almacenamiento en la memoria de un ordenador que contiene un valor que puede ser modificado (variable) durante el proceso. Ejemplo **int i = 0**; sentencia que identifica con el nombre simpolico i a un espacio de memoria que inicialmente tiene el valor entero o pero que puede ser modificado asignándole el resultado de una expresión entera.
2. Constante → asociado a un espacio de memoria que contiene un valor FIJO que no cambia ni puede ser cambiado durante el proceso o la ejecución del programa. Solo puede ser modiicado editando el programa, modificando el valor y a través de una nueva compilacion. Se pueden declarar de dos formas:
    a. Mediante la utilización de la palabra reservada *const* → **const float PI = 3.1416**;
    b. Con la directiva de preprocesador → **# define PI 3.1416**. En este caso cada aparición de el identificador PI es reemplazado por el valor 3.1416
3. Funciones → asociado a un espacio de memoria que contiene líneas de código que se especializan en realizar una acción determinada **int miFuncion(int, int**);

ValorL
Siendo int a=10, b=5; c;
La sentencia c = a+b; le asigna a c la suma de los valores de a y b. el identificador c esta situado a la izquierda (left) de la asignación, por esta condicion recibe el nombre de valor. Solo pueden ser ValorL las variables, las constantes no cumplen con esto dado que su valor, como se menciono, no se puede modificar durante la ejecución del programa, solo puede ser cambiado modificando el código y compilando nuevamente.

Declaraciones y definiciones

Una Declaracion es una construcción que especifica las propiedades de un identificador : declara lo que una palabra (identificador) "significa". Especifica la existencia y la semántica de los identificadores al compilador ya sea el tipo de dato para variables y constantes, o la firma para funciones). Declaracion asocia un nombre a un determinado tipo de dato, lo que brinda información fundamental al compilador. No hace nada más que la asociación: relacionaa una etiqueta con un tipo. Cuando esta etiqueta se "asociada" con una entidad concreta para hacer uso de las operaciones permitidas sobre esta entidad, ya sea un dato (variable o constante) o un algoritmo (una función).
El término "declaración", entonces, se contrasta con el término "definición". C es fuertemente tipado por lo que las declaraciones son imprescindibles. Si una declaración de una constante o variable especifica el valor permanente de la constante o el valor inicial de una variable, algunos

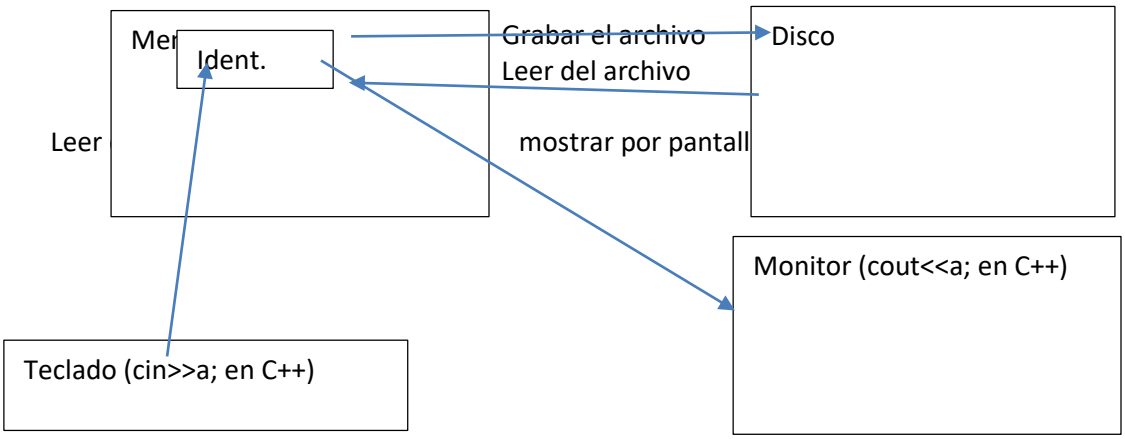
lenguajes lo llaman definición, si solo especifica el tipo declaración. De forma similar se puede pensar las funciones, la declaración es la firma y la definición la implementación.
La declaración se completa con la definición, es aquí donde se concreta la creación de la entidad. Las declaraciones pueden definir y/o referenciar, si además reserva almacenamiento a un objeto o función es una definición.

Expresiones y sentencias

Por definición una expresión es un conjunto de operadores y operandos que reducen a un valor, por ejemplo siendo a un entero con valor 5 y b otro entero con valor 10; a+b es una expresión que tine operadores (+) y operansos (a,b) en este caso reducen al valor 5 + 10, es decir 15. Por lo que a+b es una expresión, en este caso aritmética. Siguiendo con a y b y los valores detallados entonces a>b también es una expresión, en este caso el operador es de relación y es ">".el valor a que reduce esta expresión es uno de dos posibles, puede ser cierto (verdadero) o no cierto (falso) deperndiendo de los valores de a y b. Por extensión a también es una expresión, el valor es 5, lo que ocurre es que no es completa ya que arece de operadores.
Una sentencia es una acción que se ejecuta efectivamente para obtener una sentencia en c se debe finalizar con ";". Ejemplo c=a+b; es una sentencia que le asigna a c el valor de la expresión a+b y termina con un ;
Las sentencias pueden ser:

1. Simples → una única acción → asignación
    a. Interna
    b. Externa
        i. Entrada
        ii. salida
2. Estructurada → responden a un formato o estructura determinada → análisis de caso, repeticiones.
    a. Análisis de caso
        i. Simple
            1. Completo
            2. Incompleto
        ii. Compuesto
            1. Completo
            2. Incompleto
    b. Iteraciones
        i. Exactas
        ii. No excactas
            1. Precondicionales
            2. Poscondicionales
3. Compuestas→ una o mas sentencias simples, estructuradas ocombinaciones que se tratan como una unidad **{**sentencia;… ;sentencia**}**

Esquema conceptual de dispositivos



Un ejemplo
Desarrollar un programa que “lea” el nombre de una persona y lo “imprima” por pantalla

Solucion conceptual

```
Comienzo
Nombre: Cadena de caracteres // declarar la variable que va a contener el nombre
Leer(nombre) // ingresar por teclado el valor que contendrá la variable nombre
Imprimir(nombre) //mostrar por pantalla el contenido de la variable nombre
Fin
```

Solucion en un LP (C++)

```
#include <iostream>//pone a disposición elementos que necesita
using namespace STD; //orienta en la búsqueda de dispositivos de E/S
int main() {
        char nombre[10]//declara donde guardar el nombre en la memoria
        cout<<”Ingrese un nombre: “;//deriva un  mensaje a la pantalla
        cin>>nombre;//recibe un valor desde el teclado y lo almacena en el espacio reservadp
        cout<< “el nombre ingresado es : “ <<nombre;//deriva mensaje e identificsdor
        return 0;// termina el prorama
}
```