UNIVERSIDAD NACIONAL AUTÓMATA DE MÉXICO

FACULTAD DE INGENIERÍA

ESTRUCTURA DE DATOS Y ALGORITMOS I

ACTIVIDAD ASÍNCRONA #1 LUNES

PÉREZ AGUILAR ROBERTO

(10/06/2021)

Problemas: son circunstancias indeseables para uno o varios individuos, se puede resolver mediante la aplicación de un algoritmo.

Planteamiento de un problema: es la base de todo proyecto pues en él se define afina y estructura una idea formal, va de lo general a lo particular.

> **Identificación del problema**: tema o cuestión a abordar.
> **Valoración del problema**: evaluación de la importancia o relevancia del problema.
> **Formulación del problema**: se cuestiona que es lo más importante a resolver
> **Definición del problema**: se investiga los antecedentes.
> **Delimitación del problema**: se establece los aspectos que serán abordados.

Algoritmo: conjunto finito y ordenado de acciones con las que podemos resolver un determinado problema.

Análisis del enunciado del problema: es necesario para el diseño de un algoritmo por lo cual es importante comprender el alcance, identificar los datos de entrada e identificar los datos de salida.

Memoria y operaciones aritméticas y lógicas: los datos ingresan por medio de la lectura de datos y permanecen en la memoria durante el tiempo de ejecución del algoritmo. Evaluar la operación implica una operación lógica, es posible realizar aritmética. Los recursos disponibles para el diseño de algoritmos son la memoria, la posibilidad de hacer operaciones matemáticas y lógicas

Teorema de programación estructurada: los algoritmos se resuelven mediante tres estructuras de control las cuales son las siguientes

- **Estructura secuencial o de acción simple**
- **Estructura de decisión o acción condicional**
- **Estructura iterativa o acción de repetición**

Conceptos de programación: las memorias tiene la capacidad de resolver operaciones matemáticas y lógicas, se tiene que especificar de tal forma que la computadora lo comprenda.

Lenguajes de programación: son lenguajes formales que se componen de un conjunto de palabras (keywords) en inglés y reglas sintácticas y semánticas, se usan para codificar el algoritmo con ayuda del compilador se traduce a lenguaje máquina.

Codificación de un algoritmo: las acciones se codifican, en un conjunto de limas conocido como código fuente, el archivo de texto debe tener una extensión que depende del lenguaje elegido.

Bibliotecas de funciones: están incluidos en los lenguajes de programación, proporcionan funcionalidad, no hay que crear las bibliotecas solo usarlas.

Programas de computación: algoritmo codificado y compilado, el cual paso por el proceso de creación, codificación y compilación.

Entrada y salida de datos: se apoya de la consola el cual es el conjunto compuesto por un teclado y pantalla del ordenador que sirve para observar el texto, los datos de entrada es el conjunto de información que ingresan al algoritmo por la manipulación del teclado, estos pueden emitirse sobre algún dispositivo el cual representa la salida. La consola es el dispositivo de entrada y salida.

Lenguajes algorítmicos: recurso que permite describir con mayor o menor detalle los pasos que componen el algoritmo, los lenguajes de programación son lenguajes algorítmicos.

Pseudocodigo: es la mezcla de un lenguaje natural con las características semánticas y sintácticas de un lenguaje de programación, los diagramas describen los pasos de un algoritmos.

Representación gráfica de algoritmos: pueden ser representados por diagramas que ayudan a dar una visión simplificada de la lógica de un algoritmo.

Estructura secuencial: la acción se representa por un cuadro, mientras que los datos de entrada y salida se representan con trapecios.

Estructura de decisión: se representa como una casa con dos habitaciones y techo a dos aguas, el techo representa la expresión lógica que se debe evaluar, si es verdadero se ejecutan las acciones de la izquierda y si es falsa se ejecutan las acciones de la derecha.

Estructura de repetición: se representa como una caja con cabecera que indica la expresión que debe de cumplirse para seguir ejecutando las acciones de la sección principal.

Módulos o funciones: un módulo representa un algoritmo para la resolución de un problema específico, para representar las acciones se usa un paralelogramo con el nombre del módulo y una R dentro de un circulo para indicar que el modulo finalizo y regresara al programa principal.

Codificación del algoritmo usando C: todos los programas se codifican dentro de la función “main” es el encabezado del programa principal, para imprimir su usa “printf”, para incluir una librería se usa “#include”, los bloques de código se delimita por llaves, todas las líneas de código llevan “;” menos include y con “return”. Cuando aumenta la complejidad será más difícil de entender para ello se puede agregar un comentario de línea con “//” y si es de más líneas con “/* */”, el compilador no lee los comentarios

La memoria de la computadora: esta es de gran utilidad para poder usar el algoritmo dependiendo de su contenido se necesitara una mayor o menor memoria.

Máquina de turing: dispositivo que manipula símbolos sobre una cinta siguiendo una tabla de reglas, puede ser adaptada para simular la lógica de cualquier algoritmo. Puede ser considerada autómata capaz de reconocer lenguajes formales.

- **Cinta:** conjunto de celdas que contiene símbolos de algún alfabeto.
- **Cabezal:** puede leer y escribir símbolos en la cinta, también la puede mover de derecha a izquierda una celda a la vez.
- **Registro de estado:** almacena el estado de la máquina de Turing.
- **Tabla de instrucciones:** tabla infinita, llamada ocasionalmente como tabla de acción o de función de transición.

Representación como diagramas de estados: las máquinas de Turing se pueden representar mediante grafos particulares conocidos como diagramas de estados finitos. Los estados se representan con vértices etiquetados por su nombre, la transición de un estado a otro se representa por una arista, el estado inicial se caracteriza por tener una arista y el estado final se representa por vértices encerrado por una circunferencia

Arquitectura de von Neumann: describe una arquitectura de diseño para una computadora digital electrónico formada por una unidad de procesamiento con una unidad logita y registros de procesador, una unidad de control que contiene un registro de instrucciones y un contador de programa, una memoria para almacenar instrucciones y un mecanismo de entrada y salida.

El byte: es la unidad de memoria mínima de información que puede ser almacenada, esta representa 8 bits, en esta cantidad podemos almacenar un número binario de 8 dígitos

Dimensiones de los datos: en un byte de memoria podemos almacenar días, calificaciones, etc. Pero para cantidades más grandes se utilizaran 2 bytes o una mayor cantidad, en 16 bits podemos almacenar números hasta 65535

Caracteres: son representados como valores numéricos positivos, cada carácter tiene asignado un valor definido en la tabla ASCII, para representarlos se usaran los bytes de memoria, usando n bytes de memoria.

Variables: es un espacio de la memoria en el cual podemos guardar temporalmente que después utilizaremos para la ejecución, para poderla usar debemos definir un nombre y el tipo de dato, al nombre de la variable lo llamaremos identificador este no debe tener espacios ni signos de puntuación. Estos además pueden cambiar durante la ejecución del programa, para que este tenga un valor se le debe asignar un operador de asignación (=), una variable solo puede contener datos del mismo tipo.

Tipos de datos en c: a cada tipo de dato entero se le puede asignar el modificador unsigned para aplicar el bit de signo, los datos lógicos o booleanos se manejan como enteros, los valores alfanuméricos o cadenas de caracteres se representan como conjuntos de datos de tipo char.

Placeholders: también conocidos como marcadores de posición (%c, &s, %s, etc.).

Estructura de control selectiva if: este es una estructura lógica si se cumple una condición es verdadera, además se ejecutan las instrucciones del bloque que se encuentra entre las llaves de la estructura, sino se cumple se continúa con el flujo normal del programa.

Estructura selectiva if-else: aplica lo mimos que el if en la primer parte más que si la condición es falsa se ejecuta el bloque de código que está entre las llaves después de la palabra reservada else, al final de que se ejecute uno u otro código se continua con el flujo normal del programa, es posible anidar la misma estructura.

Estructura de control selectiva switch-case: evalúa la variable que se encuentra dentro del paréntesis después de la palabra reservada switch y los compara con cada valor que posee cada caso case, los tipos de datos que pueden ser evaluados son enteros, caracteres, y enumeraciones, al final de cada caso se ejecuta la instrucción break, si se omite se ejecutara el siguiente caso.

Estructura de control repetitiva do-while: es un ciclo que ejecuta el bloque de código que se encuentra dentro de las llaves y esto continúa hasta que ya no se cumpla la condición.

Estructura de control de repetición for: la estructura for se compone de 3 partes, la primera es la inicialización, la segunda es la condición, la tercera es el conjunto de operaciones que se van a realizar.

Arreglos unidimensionales y multidimensionales: se inicia con el número cero y se resta un 1, se compone del tipo de dato nombre y su tamaño, esto puede variar convirtiéndose en una matriz.

Apuntadores: nos ayuda a contener la dirección de una variable, hace referencia de su ubicación.

Define: nos ayuda a definir un valor, más bien establecer que siempre valga lo mismo y lo podemos utilizar como una macro.

Funciones: permite hacer la división de un problema grande en unas partes más pequeñas para hacer más fácil el código.

Programación: la programación se puede aplicar en las ciencias físicas, médicas, sociales, artes y humanidades, etc.

Áreas de la programación:

- **Desarrollo web:** todo lo que se puede ejecutar en el ordenador, se puede dividir en sitio web (son las páginas web informativas), aplicaciones web (es una aplicación completa y contiene una lógica compleja). El desarrollo web se puede dividir en la backend es la parte encargada del lado del servidor y frontend se encarga del lado del cliente es lo que vemos
- **Desarrollo móvil:** es crear aplicaciones de teléfono que funciona en android de Google y IOS de Apple, las aplicaciones de cada sistema uno de estos dos son aplicaciones nativas, las aplicaciones multiplataforma funciona en los dos sistemas.
- **Videojuegos:** hay diseñadores de storytelling, para desarrollarlos se puede usar Unity 3D que utiliza C# y Unreal Engine que usa C++
- **Desarrollo de aplicaciones de escritorio:** aplicaciones que se instalan directamente en la computadora sea de Windows, Linux, Mac OS, etc.
- **Sistemas operativos/embebidos:** son la capa más baja de software que se comunica con el hardware y los sistemas embebidos son aquellos que son diseñados para cubrir necesidades específicas normalmente siempre son chips.
- **Seguridad informática:** principalmente se usa Python ya que permite automatizar procesos, también es bueno conocer c, bash, SQL para las bases de datos.
- **Machine learning:** consiste en enseñarles a las computadoras por medio de una gran cantidad de datos para que se pueda crear algo, y así crear patrones para poder predecir una acciono a futuro.

- **Cloud computing:** la nube es una red mundial que permite el almacenamiento, se usan la mayoría de los lenguajes pero para automatizar procesos se puede utilizar Python.

Programación paralela: permite ejecutar más acciones en menos tiempo, ya que divide un gran problema en varios más pequeños y los resuelve al mismo tiempo, cada parte se descompone en una serie de instrucciones, las instrucciones de cada parte se ejecutan simultáneamente en diferentes procesadores, se emplea un mecanismo de control/coordinación.

Programación lógica: es un tipo de paradigma de programación dentro del paradigma de programación declarativa. Esta gira entorno del concepto de predicado y su evolución además abarca parte matemática.

Programación estructurada: orientada a mejorar la claridad, calidad y tiempo de desarrollo de un programa de computadora utilizando únicamente estructuras de selección (if y switch) e iteración (bucles for y while) es inútil la transferencia incondicional ya que el código no seria entendible.

Software propietario: es cualquier programa informático en el que los usuarios tienen limitadas las posibilidades de modificarlo

Software libre: los usuarios pueden ejecutar, distribuir, estudiar, modificar y mejorar el software. Tiene la libertad de usar el programa con cualquier propósito.

Acordeón de lenguaje C

- **Comentarios:** el comentario por línea inicia con los símbolos "//" y termina hasta donde finaliza el renglón, los comentarios por bloque inicia con "/* y finaliza con "*/" este puede abarcar varios renglones
- **Declaración de variables:** se hace por medio de la siguiente sintaxis "[modificadores] tipoDeDato identificador [= valor]. Los tipos de datos son caracteres, enteros sin punto decimal, flotantes que son números reales de precisión normal y los dobles que son reales de doble precisión. El especificador de formato nos ayuda a guardar o imprimir, para enteros en base 10 %i o %d (%li o %ld si son más largos, si es para imprimir un entero base 8 se utiliza %o y para base 16 es %x, para valor real %f, para reales de doble precisión %lf, %e para notación científica, %g redondea a 3 cifras significativas, %c un carácter y %s cadena de caracteres.
- **Identificador:** es el nombre con el que se va almacenar un tipo de dato, puede contener palabras de a-z mayúsculas y carácter de guion bajo en lugar de espacios, se recomienda la notación del camello ejemplo tipoDeDato.
- **Modificadores de alcance:** "const" impide que el valor cambie durante la ejecución del programa, "static" permanece estática en la memoria

- **Operadores:**

Operador	*Operación*	*Uso*	*Resultado*
+	Suma	125.78 + 62.5	188.28
-	Resta	65.3 - 32.33	32.97
*	Multiplicación	8.27 * 7	57.75
/	División	15 / 4	3.75
%	Módulo	4 % 2	0

Operador	*Operación*	*Uso*	*Resultado*
>>	Corrimiento a la derecha	8 >> 2	2
<<	Corrimiento a la izquierda	8 << 1	16
&	Operador AND	5 & 4	4
|	Operador OR	3 | 2	3
~	Complemento ar-1	~2	1

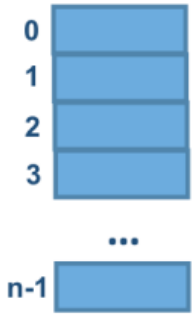
- **Moldeo o cast:** las operaciones da un tipo de dato diferente y es necesario moldearlo.
- **Expresiones lógicas:**

Operador	*Operación*	*Uso*	*Resultado*
==	Igual que	'h' == 'H'	Falso
!=	Diferente a	'a' != 'b'	Verdadero
<	Menor que	7 < 15	Verdadero
>	Mayor que	11 > 22	Falso
< =	Menor o igual	15 <= 22	Verdadero
> =	Mayor o igual	20 >= 35	Falso

- **Depuración de programas:** es cuando no falla la ejecución del programa.
- **Estructura de control selectiva if:** sintaxis if expresión_logica){ // bloque de código}, evalúa la expresión lógica si se cumple se ejecuta el bloque de código, sino se sigue el flujo normal del programa, si el bloque de código solo consta de una línea no son necesarias la llaves
- **Estructura de control selectiva if-else:** if (expresión_logica) {// bloque de código si es verdadera} else {// bloque de código si la condición es falsa}. Si la condición es verdadera se ejecuta el código de la primera llave y si es falsa se ejecuta el segundo código de la segunda llave, es posible anidar la misma estructura.
- **Estructura de control selectiva switch-case:** switch (opcion_a_evaluar) {case valor1: /* Código a ejecutar*/ break; case valor2: /* Código a ejecutar*/ break;… case valorN: /* Código a ejecutar*/ break; default: /* Código a ejecutar*/}. Evalúa y compara de acuerdo a cada case y ejecuta su código, es importante usar break al finalizar cada case ya que si no se hace ejecutara el siguiente código.
- **Enumeración:** enum identificador(VALOR1, VALOR 2, …VALORN);, enum boolean (FALSE, TRUE); esto nos puede ayudar en las estructuras de control selectivas
- **Estructura de control repetitiva while:** while (expresión_lógica) { // Bloque de código a repetir mientras que la expresión lógica sea verdadera.}, primero valida la expresión y si se cumple ejecutara el código hasta que no sea válida.
- **Estructura repetitiva do-while:** do { /* Bloque de código que se ejecuta por lo menos una vez y se repite mientras la expresión lógica sea verdadera. */ } while (expresión_lógica); ejecuta el primer bloque de código hasta que ya no sea válido.
- **Estructura de control de repetición for:** for (inicialización; expresión_lógica; operaciones por iteración) {/* Bloque de código a ejecutar

*/}, primero se establece el valor, después la condición y por último la operación, deja de ejcutarse hasta cumplir la condición.

- **Define:** #define nombre valor, es una macro que sirve para el definir un valor
- **Break:** provoca que el ciclo se cierre inmediatamente.
- **Arreglos unidimensionales:**



la primera localidad las compone el número 0 y la ultima la compone el elemento n-1, La sintaxis para un la siguiente: tipoDeDato nombre [tamaño], también se puede usar las estructura de control repetitiva como son el for y while.

- **Apuntadores:** es una variable que contiene la dirección de una variable, la sintaxis es la siguiente TipoDeDato *apuntador, variable; apuntador = &variable;
- **Arreglos multidimensionales:** su estructura es la siguiente tipoDato nombre [tamaño] [tamaño]… [tamaño]; la primera dimensión corresponde a los renglones, la segundo a las columnas, la tercera al plano. Se pueden recoger arreglos con apuntadores.
- **Funciones:** valorRetorno nombre (parámetros) {// bloque de código de la función}, nos permite dividir un problemas en varios bloques de código.
- **Abrir y cerrar archivos:** *FILE fopen (char *nombre_archivo, char *modo); int fclose (FILE *apArch); r: Abre un archivo de texto para lectura. w: Crea un archivo de texto para escritura. a: Abre un archivo de texto para añadir. r+: Abre un archivo de texto para lectura / escritura. w+: Crea un archivo de texto para lectura / escritura. a+: Añade o crea un archivo de texto para lectura / escritura. rb: Abre un archivo en modo lectura y binario. wb: Crea un archivo en modo escritura y binario.
- **Funciones fgets y fputs:** Las funciones fgets() y fputs() pueden leer y escribir, cadenas sobre los archivos.
- **Funciones fscanf y fprintf:** fprintf() y fscanf() se comportan exactamente como printf() (imprimir) y scanf() (leer), excepto que operan sobre archivo. int fprintf(FILE *apArch, char *formato, ...); int fscanf(FILE *apArch, char *formato, ...);.

- **Funciones fread y fwrite:** permiten trabajar con elementos de longitud conocida. fread permite leer uno o varios elementos de la misma longitud a partir de una dirección de memoria determinada (apuntador). int fread(void *ap, size_t tam, size_t nelem, FILE *archivo). fwrite permite escribir hacia un archivo uno o varios elementos de la misma longitud almacenados a partir de una dirección de memoria determinada. int fwrite(void *ap, size_t tam, size_t nelem, FILE *archivo).