JORGE ALVARADO MENDOZA
21414141

14/05/2021

UNIDAD #1

MODELO DE REPRESENTACION DE ESTRUCTURA DE DATOS

1: LA ABSTRACCION

ES UNA DE LAS HERRAMIENTAS QUE NOS AYUDA A SOLUCIONAR UN PROBLEMA, ES UN MECANISMO FUNDAMENTAL PARA LA COMPRESION DE PROBLEMAS Y FENOMENOS QUE POSEEN UNA GRAN CANTIDAD DE DETALLES SU IDEA PRINCIPAL CONSISTE EN MANEJAR UN PROBLEMA, FENOMENO, OBJETO, TEMA O IDEA COMO CONCEPTO GENERAL, SIN CONSIDERAR LA GRAN CANTIDAD DE DETALLES QUE ESTOS PUEDEN TENER.

EL PROCESO DE ABSTRACCION PRESENTA DOS ASPECTO COMPLEMENTARIO

1. DESTACAR LOS ASPECTOS RELEVANTES DEL OBJETO
2. IGNORAR LOS ASPECTOS IRRELEVANTES DEL MISMO (LA IRRELEVANCIA DEPENDE DEL NIVEL DE ABSTRACCION, YA QUE SI SE PASA NIVELES MAS CONCRETOS, ES POSIBLE QUE CIERTOS ASPECTOS PASEN HACER RELEVANTES

LOS DIFERENTES TIPOS DE ABSTRACCION

1. ABSTRACCION FUNCIONAL SON UNIDADES, LIBRERIAS PAQUETES QUE DETERMINAN EL COMPORTAMIENTO Y USO DE DETERMINADOS MODULOS

2. ABSTRACCION DE DATOS

- TIPO DE DATOS PROPORCIONADO POR LOS LENGUAJES DE ALTO NIVEL LA PRESENTACION USADA ES INVISIBLE AL PROGRAMADOR, AL CUAL SOLO SE PERMITE VER LAOPERACIONES PRE DEFINIDAS PARA CADA TIPO.
- DEFINIDOS POR EL PROGRAMADOR QUE POSIBILITAN LA DEFINICION DE VALORES DE DATOS MAS CERCANOS AL PROBLEMA QUE SE PRESENTE RESOLVER
- TDA PARA LA DEFINICION Y REPRESENTACION DE TIPOS DE DATOS (VALORES + OPERACIONES), JUNTO CON SUS PROPIEDADES.
- OBJETOS SON TDA A LOS QUE AÑADE PROPIEDADES DE REUTILIZACION Y DE COMPARTICION DE CODIGO

3. ABSTRACCIÓN DE CONTROL SON EL PROCEDIMIENTO QUE AL INVOCARLOS MEDIANTE UN NOMBRE SE DESTACA QUE HACE LA FUNCION Y SE IGNORA COMO LO HACE. EL USUARIO SOLO NECESITA CONOCER LA ESPECIFICACION DE LA ABSTRACCION (EL QUE) Y PUEDE IGNORAR EL RESTO DE LOS DETALLES (EL COMO)

SI PROFUNDIZAMOS MAS AL MUNDO DE LA PROGRAMACION Y SUS CONCEPTOS, EXISTEN DOS DE ESTOS CONCEPTOS QUE NO SE DEBEN CONFUNDIR ELLOS SON: TIPOS DE DATOS Y ESTRUCTURAS DE DATOS

UN TIPO DE DATO, ES EN LENGUAJE DE PROGRAMACION DEFINE UN CONJUNTO DE VALORES QUE UNA DETERMINADA VARIABLE PUEDE TOMAR ASI COMO LAS OPERACIONES BASICAS SOBRE DICHO CONJUNTO.

AHORA VEAMOS COMO SE VA RELACIONANDO ESTOS CONCEPTOS. LOS TIPOS DE DATOS CONSTITUYEN UN PRIMER NIVEL DE ABSTRACCION YA QUE NO SE TIENE EN CUENTA COMO SE IMPLEMENTAN O SE REPRESENTAN O SE REPRESENTAN REALMENTE LA INFORMACION SOBRE LA MEMORIA DE LA MAQUINA PARA EL USUARIO EL PROCESO DE IMPLEMENTACION O REPRESENTACION ES INVISIBLE

2. ESTRUCTURAS DE DATOS

COMENZEMOS POR ESTABLECER QUE ESTRUCTURA DE DATOS ES LA FORMA DE ORGANIZAR UN CONJUTO DE DATOS CON EL OBJETIVO DE FACILITAR SU MANIPULACION UN DATO ES LA MISMA INFORMACION QUE SE TIENE EN UN SISTEMA.

POR LO QUE CUANDO SE DEFINA UNA ESTRUCTURA DE DATOS ESTAREMOS DEFINIENDO LA ORGANIZACION EN INTERRELACION DE ESTOS Y UN CONJUNTO DE OPERACIONES MINIMAS QUE SE PUEDEN REALIZAR SOBRE ELLOS COMO SER:

- ALTA, ADICIONAR UN NUEVO VALOR A LA ESTRUCTURA
- BAJA, BORRAR, UN VALOR DE LA ESTRUCTURA
- BUSQUEDA, ENCONTRAR UN DETERMINADO VALOR EN LA ESTRUCTURA PARA REALIZAR UNA OPERACION CON ESTE VALOR

OTRAS OPERACIONES QUE SE PUEDEN REALIZAR SON:

- ORDENAMIENTO. DE LOS ELEMENTOS PERTENECIENTES A LA ESTRUCTURA
- APAREO, DADOS DOS ESTRUCTURAS ORIGINAR UNA NUEVA ORDENADA Y QUE CONTENGA A LAS APAREADAS
- OTRAS DEFINIDAS DE ACUERDO A LA CARACTERISTICA DE LA ESTRUCTURA

DE TAL FORMA QUE PARA RESOLVER UN PROBLEMA INTERVIENEN LOS SIGUIENTES ELEMENTOS PROGRAMA ALGORITMO Y ESTRUCTURA DE DATOS

PROBLEMA $\xrightarrow{\text{ALGORITMOS} + \text{ESTRUCTURA DE DATOS}}$ PROGRAMA

PROBLEMA CONJUNTO DE HECHOS / CIRCUNSTANCIAS QUE DIFICULTAN LA CONSECUSION DE ALGUN FIN

ALGORITMO CONJUNTO DE REGLAS FINITAS E INAMBIGUO

ESTRUCTURA DE DATOS DISPOSICION EN MEMORIA DE LA INFORMACION

... ALGORITMOS + ESTRUCTURA DE DATOS

COMO YA ES CONOCIDO POR USTEDES POR LO GENERAL TENEMOS QUE TRATAR CON DATOS SIMPLES (ENTEROS, REALES, BOLEANOS, ETC) QUE POR SI SOLOS NO NOS DICEN NADA, M NOS SIRVEN MUCHO, ES NECESARIO TRATAR, CON ESTRUCTURAS DE DATOS ADECUADAS A CADA NECESIDAD

LAS ESTRUCTURA DE DATOS SON UNA COLECCION DE DATO CUYA ORGANIZACION SE CARACTERIZA POR LAS FUNCIONES DE ACCESO QUE SE USAN PARA ALMACENAR Y ACCEDER A ELEMENTOS INDIVIDUALES DE DATOS.

CADA ESTRUCTURAS OFRECE VENTAJAS Y DESVENTAJAS EN RELACION A LA SIMPLICIDAD Y EFICIENCIA PARA LA REALIZACION DE CADA OPERACION DE ESTA FORMA, LA ELECCION DE LA ESTRUCTURA DE DATOS APROPIADA

PARA CADA PROBLEMA DEBEMOS LOS FACTORES COMO LA FRECUENCIA Y EL ORDEN EN QUE SE REALIZA CADA OPERACION SOBRE LOS DATOS

EJEMPLO: PARTIENDO DE LA LOGICA DE QUE CUANDO APRENDEMOS A PROGRAMAR NOS PIDEN GUARDAR EL NOMBRE, SUELDO, CARNET DE UNA PERSONA NORMALMENTE EN EL CODIGO DECLARARIAMOS LAS VARIABLES NOM DEL TIPO CADENA, SUELDO DEL TIPO REAL Y CARNET DE TIPO CADENA, SIN EMBARGO EL PROBLEMA SE PONE COMPLEJO SI QUEREMOS GUARDAR LA INFORMACION DE 150 PERSONAS LO CUAL NOS LLEVA RAPIDAMENTE A LA IMPOSIBILIDAD DE DECLARAR 450 VARIABLES EN NUESTRO CODIGO POR LO QUE TENDREMOS QUE RECURRIR AL USO DE ESTRUCTURAS QUE PERMITAN MANEJAR LA CANTIDAD DE INFORMACION ADECUADAMENTE

CLASIFICACION DE LOS ESTRUCTURAS DE DATOS

LAS ESTRUCTURAS DE DATOS SE CLASIFICAN EN DOS, CONSIDERAMOS LA EXISTENCIA DE TIPOS DE DATOS BASICA Y ESTABLECIDOS Y LOS NUEVOS POR CREAR

- LAS ESTRUCTURAS DE DATOS PRIMITIVAS SON LOS TIPOS DE DATOS FUNDAMENTALES QUE SON COMPATIBLES CON UNA PROGRAMACION, ALGUNOS TIPOS DE DATOS BASICOS SON ENTEROS, REALES, CARACTERES, Y BOLEANOS ETC.
- LAS ESTRUCTURAS DE DATO [illegible] SON AQUELLAS ESTRUCTURAS DE DATOS QUE SE CREAN UTILIZANDO DATOS PRIMITIVOS. LOS EJEMPLOS DE TALES ESTRUCTURAS DE DATOS INCLUYEN LISTAS, PILAS, CONJUNTO ETC. SE PUEDE CLASIFICAR EN DOS CATEGORIAS: LINEAL Y NO LINEAL
  - LINEAL: SI LOS ELEMENTOS DE UNA ESTRUCTURA DE DATOS SE ALMACENAN EN UN ORDEN LINEAL O SECUENCIAL, ENTONCES PERTENECE A LA CATEGORIA LINEAL
  - NO LINEAL: SI LOS ELEMENTOS DE UNA ESTRUCTURAS DE DATOS NO SE ALMACENAN EN UN ORDEN SECUENCIAL, ENTONCES ES UNA ESTRUCTURA DE DATOS NO LINEAL

3.: TIPO DE DATOS ABSTRACTO

EL CONCEPTO DE TIPO DE DATO ABSTRACTO (TDA, ABSTRACT DATA TYPES), FUE PROPUESTO POR PRIMERA VEZ HACIA 1974 POR JOHN GUTTAG Y OTROS, PERO NO FUE HASTA 1975 QUE POR PRIMERA VEZ BARBARA LISKOV LO PROPUSO PARA EL LENGUAJE CLU.

PRIMERA MUJER EN OBTENER UN DOCTORADO EN CIENCIAS DE LA COMPUTACION EN LOS ESTADOS UNIDOS (STANFORD 1968)

DEFINICION:

CON MUCHA FRECUENCIA SE UTILIZAN LOS TERMINOS TDA Y ABSTRACCION DE DATOS DE MANERA EQUIVALENTE Y ESTO ES DEBIDO A LA SIMILITUD E INTERDEPENDENCIA DE AMBOS, SIN EMBARGO ES IMPORTANTE DEFINIR POR SEPARADO LOS DOS CONCEPTOS

COMO YA SE MENCIONO, LOS LEGUAJES DE PROGRAMACION ORIENTADO A OBJETOS SON LENGUAJES FORMADOS POR DIFERENTES METODOS O FUNCIONES Y QUE SON LLAMADOS EN EL ORDEN EN QUE EL PROGRAMA LO REQUIERE O EL USUARIO LO DESEA. LA ABSTRACCION DE DATOS CONSISTE EN OCULTAR LAS CARACTERISTICAS DE UN OBJETOS Y OBTAR LAS DE MANERA SOLAMENTE UTILIZAMOS EL NOMBRE DEL OBJETO EN NUESTRO PROGRAMA. ESTO ES SIMILAR A UNA SITUACION DE LA VIDA COTIDIANA. CUANDO YO DIGO LA PALABRA "PERRO", USTED NO NECESITA QUE YO LE DIGA LO QUE HACE EL PERRO, USTED YA SABE LA FORMA QUE TIENE UN PERRO Y TAMBIEN SABE QUE LOS PERROS LADRAN DE MANERA QUE ABSTRAEMOS TODAS LAS CARACTERISTICAS DE TODOS LOS PERROS EN UN SOLO TERMINO, AL CUAL LLAMO "PERRO". A ESTO SE LE LLAMA "ABSTRACCION" Y ES UN CONCEPTO MUY UTIL EN LA PROGRAMACION YA QUE UN USUARIO NO NECESITA MENCIONAR TODAS LAS CARACTERISTICAS Y FUNCIONES DE UN OBJETO CADA VEZ QUE ESTE SE UTILICE SINO SON DECLARADAS POR SEPARADO EN EL PROGRAMA Y SIMPLEMENTE SE UTILIZA EL TERMINO ABSTRACTO ("PERRO") PARA MENCIONARLOS

EN EL EJEMPLO ANTERIOR, "PERRO" ES UN TIPO DE DATO ABSTRACTO Y TODO EL PROCESO DE DEFINIRLO, IMPLEMENTARLO Y MENCIONARLOS ES A LO QUE LLAMAMOS ABSTRACCION DE DATOS.

VAMOS A PONER UN EJEMPLO REAL DE LA PROGRAMACION, SUPONGAMOS QUE EN ALGUN LENGUAJE DE PROGRAMACION ORIENTADO A OBJETOS UN PEQUEÑO PROGRAMA SACA EL AREA DEL RECTANGULO DE LAS DIMENCIONES DE UN USUARIO DECIDA. PENSEMOS TAMBIEN QUE EL USUARIO PROBABLEMENTE QUIERA SABER EL AREA DE VARIOS RECTANGULOS, SERIA MUY TEDIOSO PARA EL PROGRAMADOR DEFINIR LA MULTIPLICACION DE "BASE" POR "ALTURA" VARIAS VECES EN EL PROGRAMA, ADEMAS QUE LIMITARIA AL USUARIO A SACAR UN NUMERO DETERMINADO DE AREAS POR ELLO EL PROGRAMA PUEDE CREAR UNA FUNCION DENOMINADA AREA, LA CUAL VA SER LLAMADA EL NUMERO DE VECES QUE SEAN NECESITADAS POR EL USUARIO ASI EL PROGRAMADOR SE EVITA MUCHO TRABAJO, EL PROGRAMA RESULTA MAS RAPIDO MAS EFICIENTE Y DE MENOR LONGITUD PARA LOGRAR ESTO SE CREA EL METODO AREA DE UNA MANERA SEPARADA DE LA INTERFAZ GRAFICA PRESENTADA AL USUARIO Y SE ESTIPULA AHI LA OPERACION A REALIZAR, DEVOLVIENDO EL VALOR DE LA MULTIPLICACION. EN EL METODO PRINCIPAL SOLAMENTE SE LLAMA EN LA FUNCION AREA Y EL PROGRAMA HACE EL RESTO

SEPARACION DE LA INTERFAZ E IMPLEMENTACION

CUANDO SE USA EN UN PROGRAMA DE COMPUTACION, UN TDA ES REPRESENTADO POR SU INTERFAZ, LA CUAL SIRVE COMO CUBIERTA A LA CORRESPONDIENTE IMPLEMENTACION. LOS USUARIOS DE UN TDA TIENEN QUE PREOCUPARSE POR LA INTERFAZ PERO NO CON LA IMPLEMENTACION YA QUE ESTE PUEDE CAMBIAR EL TIEMPO Y AFECTAR A LOS PROGRAMAS USANDO QUE USAN EL TDA. ESTO SE BASA EN EL CONCEPTO DE OCULTACION DE INFORMACION, UNA PROTECCION PARA EL PROGRAMA DE DECISIONES DE DISEÑO QUE SON OBJETOS DE CAMBIO

LA SOLIDEZ DE UN TDA REPOSA EN LA IDEA DE QUE LA IMPLEMENTACION ESTA ESCONDIDA AL USUARIO SOLO LA INTERFAZ ES PUBLICA. ESTO SIGNIFICA QUE EL TDA PUEDE SER IMPLEMENTADO DE DIFERENTES FORMAS PERO MIENTRAS SE MANTENGA CONSISTENTE CON LA INTERFAZ, LOS PROGRAMAS QUE LO USAN NO SE VEN AFECTADOS

CARACTERIZACION

UN TDA ESTA CARACTERIZADO POR UN CONJUNTO DE OPERACIONES (FUNCIONES) AL CUAL LE DENOMINARON USUALMENTE COMO INTERFAZ PUBLICA Y REPRESENTAN EL COMPORTAMIENTO DE TDA, MIENTRAS QUE LA IMPLEMENTACION COMO LA PARTE PRIVADA DEL TDA ESTA OCULTA EN EL PROGRAMA CLIENTE QUE LO USA

TODOS LOS LENGUAJES DE ALTO NIVEL TIENEN PREDEFINIDOS Y ESTOS TIENEN SUS INTERFACES PUBLICAS QUE INCLUYEN LAS OPERACIONES COMO LA +, -, *, etc. NO SE NECESITA CONOCER COMO ACTUAN TALES OPERACIONES SOBRE LA REPRESENTACION INTERNA DE LOS TIPOS DEFINIDOS, QUE ADEMAS SUELE SER UNA IMPLEMENTACION BASTANTE DEPENDIENTE DE LA MAQUINA SOBRE LA QUE TRABAJE EL COMPILADOR, LO INTERESANTE ES QUE LOS LENGUAJES ACTUALES NOS VAN A PERMITIR AMPLIAR LOS TDA PREDEFINIDOS CON OTROS QUE SERAN DEFINIDOS POR EL PROPIO PROGRAMADOR PARA ADECUAR ASI LOS TIPOS DE DATOS A LAS NECESIDADES DE LOS PROGRAMAS

NOTESE QUE CUANDO HABLEMOS DE UN TDA NO HAREMOS NINGUNA ILUSION AL TIPO DE LOS ELEMENTOS SINO TAN SOLO A LA FORMA EN QUE ESTAN DISPUESTOS ESTOS ELEMENTOS. SOLO NOS INTERESA LA ESTRUCTURA QUE SOPORTA LA INFORMACION Y SUS OPERACIONES. PARA DETERMINAR EL COMPORTAMIENTO ESTRUCTURAL BASTA CON OBSERVAR LA CONDUCTA QUE SE SEGUIRA LOS DATOS.

CARACTERIZAMOS ENTONCES LOS TDA. UN TDA TENDRA UNA PARTE QUE SERA VISIBLE AL USUARIO LA CUAL HAY QUE PROTEGER Y QUE SE PUEDE DECIR QUE ES IRRELEVANTE PARA EL USO DEL USUARIO Y ESTA CONSTITUIDA TANTO POR LA MAQUINARIA ALGORITMICA QUE IMPLEMENTE LA SEMANTICA DE LAS OPERACIONES COMO POR LOS DATOS QUE SIRVAN DE ENLACE ENTRE LOS ELEMENTOS DEL TDA, ES DECIR INFORMACION INTERNA NECESARIA PARA LA IMPLEMENTACION QUE SE ESTA HACIENDO PARA ESE COMPORTAMIENTO DEL TDA, RESUMIENDO PODEMOS DECIR QUE TANTO LA IMPLEMENTACION DE LAS OPERACIONES COMO LOS ELEMENTOS INTERNOS DEL TDA SERAN PRIVADO DEL ACCESO EXTERNO Y OCULTOS A CUALQUIER OTRO NIVEL.

UN TDA REPRESENTA UNA ABSTRACCION:

- SE DESTACAN LOS DETALLES (NORMALMENTE POCOS) DE LA ESPECIFICACION (EL QUE)
- SE OCULTAN LOS DETALLES (CASI SIEMPRE NUMEROSO) DE LA IMPLEMENTACION (EL COMO)

ESPECIFICACION DE UN TDA:

LA ESPECIFICACION DE UN TDA SE PUEDE REPRESENTAR DE DOS FORMAS DE LAS CUALES SON:

- ESPECIFICACION NO FORMAL/INFORMAL
- ESPECIFICACION FORMAL

ESPECIFICACION NO FORMAL: (USANDO LENGUAJE NATURAL) PARA LA ESPECIFICACION DE UN TIPO DE DATO ABSTRACTO INFORMAL SE DEBE SEGUIR EL SIGUIENTE ESQUEMA:

1. ELEMENTOS QUE CONFORMAN LA ESTRUCTURA DE DATOS
   - TIPO DE DATOS: NOMBRE DEL TIPO
   - VALORES: DESCRIPCION DE LOS POSIBLES VALORES
   - OPERACIONES: CITA DE CADA OPERACION
2. DESCRIPCION DE LAS OPERACIONES DE LA ESTRUCTURA:

SE DESCRIBE CADA UNA DE LAS OPERACIONES SOBRE EL TDA:

NOMBRE DE LA OPERACION
DESCRIPCION BREVE DE SU UTILIDAD
DATOS DE ENTRADA A LA OPERACION
DATOS QUE GENERA COMO SALIDA LA OPERACION
PRECONDICION CONDICION QUE DEBERA CUMPLIRSE ANTES DE UTILIZAR LA OPERACION
POSCONDICION CONDICION EN QUE QUEDA EL TDA DESPUES DE EJECUTAR LA OPERACION

ESPECIFICACION FORMAL (USANDO PSEUDO CODIGO O UN LENGUAJE DE PROGRAMACION)
Y ESTA CONSTITUIDO POR LOS SIGUIENTES PUNTOS.

1. TIPO: NOMBRE DEL TDD
2. SINTAXIS FORMA DE LAS OPERACIONES
   NOMBRE DE LAS OPERACIONES (ARGUMENTOS) → RESULTADO
3. SEMANTICA SIGNIFICADO DE LAS OPERACIONES
   NOMBRE DE LAS OPERACIONES (VALORES PARTICULARES) → EXPRESION RESULTANTE

PARA MAYOR COMPRENSION SE PRESENTA PARTE DEL TDD CADENA

NO FORMAL	FORMAL // PSEUDO CODIGO LENGUAJE

1. ELEMENTOS QUE CONFORMAN LA ESTRUCTURA DE DATOS.
TDD CADENAS (VALORES TODOS LOS CARACTERES ALFABETICOS VISIBLES, OPERACIONES CREAR, ADICIONAR CARACTER, LLENA, INSERTAR CARACTER, ELIMINAR CARACTER, NUMEROCARACTERES)
2: DESCRIPCION DE LAS OPERACIONES DE LA ESTRUCTURA
CREAR
UTILIDAD: SIRVE PARA INICIALIZAR LA CADENA EN VACIO
ENTRADA CADENA S QUE SERA INICIALIZADA
SALIDA CADENA S INICIALIZADA
PRECONDICION NINGUNA
POSCONDICION LA CADENA S TIENE 0 CARACTERES
ADICIONARCARACTER
UTILIDAD SIRVE PARA AGREGAR UN CARACTER AL FINAL DE UNA CADENA
ENTRADA CADENA S Y CARACTER L QUE SE AÑADE A LA CADENA S
SALIDA CADENA S MODIFICADA
PRECONDICION LA CADENA S TIENE EL CARACTER L QUE QUEDA AL EXTREMO DERECHO DE LA CADENA
LLENA
UTILIDAD SIRVE PARA VERIFICAR SI UNA CADENA ESTA LLENA O NO
ENTRADA CADENA S QUE SERA VERIFICADA
SALIDA SI LONGITUD DE S >= 80 VERDAD, SINO FALSO
PRECONDICION: NINGUNA
POSCONDICION: NINGUNA (PUES LA CADENA S NO SE MODIFICA)
	OP:
	TIPO: CADENA
	SINTAXIS:
	CREAR (S)
	ADICIONARCARACTER (S,L)
	LLENA(S)
	INSERTARCARACTER (S,P,L)
	ELIMINAR CARACTER (S,P)
	NUMEROCARACTERES (S)
	SEMANTICA
	PARA TODO L QUE PERTENECE A CADENA S
	LLENA (CREAR()) → FALSO
	NUMERO CARACTERES (CREAR()) → 0
	OPCION 2
	CLASS CADENA{
	PRIVATE
	// DATOS MIEMBRO DE LA CLASE
	CHAR S[79];
	PUBLIC:
	// FUNCIONES MIEMBRO
	VOID ADICIONARCARACTER (CHAR L);
	VOID ELIMINAR CARACTER (INT P)
	VOID INSERTARCARACTER (INT P, CHAR L)
	BOOL LLENA()

4.- FORMA DE IMPLEMENTACION DEL TDA.

COMO SE PUDO APRECIAR EN LA LECTURA DEL PRESENTE DOCUMENTO SE HA EVIDENCIADO QUE LA IMPLEMENTACION DE LOS TDS ES INDEPENDIENTE DE LA ESPECIFICACION DE LOS MISMOS LO CUAL RATIFICA LO INDICADO ANTERIORMENTE "UN TDA PUEDE SER IMPLEMENTADO DE VARIAS FORMAS SIN QUE ESTO AFECTE A LAS APLICACIONES QUE USEN EL MISMO" ES ASI QUE LAS CUATRO FORMAS DE IMPLEMENTACION QUE SE DETALLARAN A CONTINUACION NO SON LAS UNICAS

4.1 MODELO ESTATICO

ESTA FORMA DE IMPLEMENTACION SE REFIERE A QUE TODO EL USO DEL RECURSO MEMORIA EN LA IMPLEMENTACION DE UN TDA SERA SIMPLEMENTE DEFINIDO ANTES DE SU EJECUCION LO CUAL NO NECESARIAMENTE NO LLEVA A USAR TIPOS DE DATOS BASICOS COMO SEAN ENTEROS REALES, CADENAS VECTORES, MATRICES

4.2. MODELO DINAMICO.

EN ESTA FORMA DE IMPLEMENTACION SE PRIORIZA EL USO RACIONAL DE LA MEMORIA LO CUAL SIGNIFICA QUE EN LA MEDIDA QUE SEA NECESARIO SE SOLICITARA ESPACIO DE LA MEMORIA, PARA LO CUAL SE SUGIERE MANEJAR EL MANEJO DE PUNTEROS

4.3 MODELO SIMULADO

A DIFERENCIA DE LAS TRES FORMAS DE IMPLEMENTACION ANTERIORMENTE CITADAS CABE ACLARAR QUE LO PLANTEADO EN ESTE PUNTO ES UNICA Y EXCLUSIVAMENTE DE CARACTER ACADEMICO PARA TENER MEJOR COMPRESION SOBRE LOS PUNTEROS POR LO QUE SE PROPONE CREAR UN TDA DENOMINADO CSMEMORIA CUYO OBJETIVO SERA SIMULAR EL COMPORTAMIENTO DE LA MEMORIA CON EL USO DE VECTORES

4.4. MODELO PERSISTENTE:

EL MODELO PERSISTENTE EN EL TIEMPO ES LA FORMA MAS USUAL QUE SE TIENE QUE HACER QUE LA IMPLEMENTACION DE UN TDA SEA PERSISTENTE EN EL TIEMPO, ESTO SIGNIFICA QUE SU COMPORTAMIENTO PODRA ESTAR EXENTO DEL USO EXCLUSIVO DE LA MEMORIA TRASLADANDO LAS ACCIONES DE SUS METODOS A GUARDAR LA INFORMACION EN ARCHIVOS.

https://youtu.be/CNvHXSOBPvQ

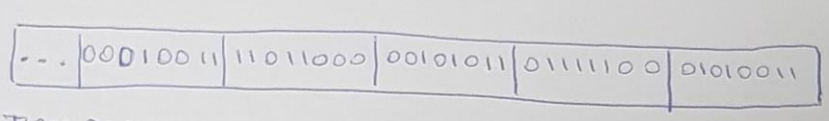
CONSIDERANDO QUE EXISTE BIBLIOGRAFIA SOBRE COMO UTILIZAR LOS MODELOS DE IMPLEMENTACION (PERSISTENTE Y ESTATICO Y DINAMICO) A CONTINUACION DESARROLLAREMOS EL MODELO SIMULADO

CONCEPTOS BASICOS

COMO FUNCIONA LA MEMORIA

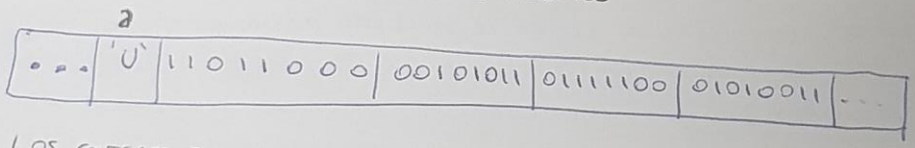
TODAS LAS VARIABLES DE UN PROGRAMA TIENE ASOCIADO UN LUGAR EN LA MEMORIA DEL COMPUTADOR

LA MEMORIA PUEDE SER VISTO COMO UN GRAN ARREGLO DE BITS, UN BIT ES LA UNIDAD BASICA DE INFORMACION QUE SE PUEDE REPRESENTAR EN UN COMPUTADOR Y PUEDE TENER LOS VALORES 0 o 1

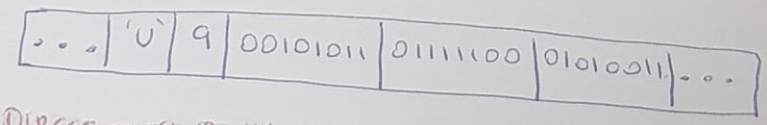


TIPOS DE DATOS

EL TAMAÑO DE UN VALOR DE TIPO CHAR ES DE 1 BYTE Y EL SIGNIFICADO DE LOS BITS ESTA DETERMINADO USANDO LA CODIFICACION ASCII. ES ASI QUE SI DEFINIMOS LA VARIABLE DEL TIPO CHAR Y LUEGO a='U' LA MEMORIA TENDRIA EN BINARIO EL CARACTER 'U' SEGUN EL SIGUIENTE GRAFICO



LOS ENTEROS SON ALMACENADOS EN SU REPRESENTACION BINARIA. LA MANERA MAS COMUN DE REPRESENTAR LOS ENTEROS NEGATIVOS ES EL COMPLEMENTO A DOS, SIENDO ASI CONSIDERADO QUE DEFINIMOS UN ESPACIO DE MEMORIA LLAMADO b DE TIPO ENTERO Y DECIMOS b=9 TENDRIAMOS LO SIGUIENTE CONSIDERANDO QUE LOS VALORES DE 'U' Y 9 ESTAN EN BINARIO.



DIRECCIONES DE MEMORIA

TODOS LOS BYTES EN LA MEMORIA TIENEN UNA DIRECCION, QUE NO ES MAS QUE UN INDICE CORRELATIVO.

POR CONVENIENCIA, LAS DIRECCIONES DE MEMORIA SUELEN ESCRIBIRSE EN NOTACION HEXADECIMAL, PERO NO HAY QUE ESPANTARSE: SE TRATA SIMPLEMENTE DE UN NUMERO ENTERO.

DIRECCION	VALOR
	. .
0x fle568	00010011
0x fle569	11011000
0x fle56a	00101011
0x fle56b	01111100
0x fle56c	01010011

SIN EMBARGO SI TENEMOS QUE GUARDAR LOS DATOS DE LA VARIABLE Z QUE TIENE DOS CAMPO, A DE TIPO CARACTER Y B DE TIPO NUMERICO, ENTONCES NUESTRO GRAFICO ANTERIOR DEBERIA QUEDAR DE LA SIGUIENTE MANERA SI DECIMOS Z.a='è' Y Z.b=9 SOLO POR EFECTOS UNICOS DE IMPLEMENTAR EL TDA SMEMORIA

DIRECCION	VALOR	ID	LINK
	. . .		NULL
0x fle568	'è'	a	0x fle56b
0x fle568	11011000		NULL
0x fle562	00101011		NULL
0x fle56b	9	b	NULL
0x fle56c	01010011		NULL
	. . .		

ES ASI QUE PARA LA IMPLEMENTACION DE NUESTRO TDA SMEMORIA UTILIZAREMOS UN VECTOR DONDE EN CADA CASILLA (DIRECCION) GUARDE DATO, ID, LINK PARA LOS SIGUIENTES FINES:

DATO: SERA EL DATO A GUARDAR EN LA MEMORIA
ID : SERA EL NOMBRE DEL ESPACIO DE MEMORIA
LINK : SERA LA DIRECCION DE OTRO ESPACIO DE MEMORIA

(SE ESTABLE QUE EL SIMULOR PLANTEADO SOLO MANIPULARA DATOS ENTEROS EN VALOR, POR LO QUE NO SE INCREMENTARA UN CAMPO PARA DETERMINAR EL TIPO DE DATO GUARDADO)

DESCRIPCION NO FORMAL DEL TDD SMEMORIA

1- ELEMENTOS QUE CONFORMARAN ESTRUCTURA

TDD SMEMORIA (VALORES TODO LOS NUMERO ENTEROS, OPERACIONES, CREAR, NEW_ESPACIO_DELETE_ESPACIO, PONER_DATO, OBTENER_DATO, DIR_LIBRE, ESPACIO_OCUPADO, ESPACIO_DISPONIBLE)

2. DESCRIPCION DE LAS OPERACIONES DE LA ESTRUCTURA

CREAR

UTILIDAD SIRVE PARA INICIALIZAR LOS ESPACIOS DE MEMORIA

ENTRADA ESPACIOS DE MEMORIA SIN RELACION

SALIDA ESPACIOS DE MEMORIA RELACIONADOS ENTRE SI

PRECONDICION NINGUNA

POSCONDICION MEMORIA LISTA PARA USARSE

NUEVO_ESPACIO

UTILIDAD: SOLICITAR N ESPACIO DE MEMORIA CON SU DEBIDA IDENTIFICACION

ENTRADA: CODENA CON LOS IDENTIFICADORES REQUERIDOS

SALIDA: DIRECCION DE MEMORIA A PARTIR DE DONDE SE PODRA GUARDAR INFORMACION PARA LOS IDENTIFICADORES REFERIDOS

PRECONDICION: MEMORIA INICIALIZADA

POSCONDICION: MEMORIA DISIMINUIDO EN SU CAPACIDAD DE DIRECCIONES LIBRES PARA TRABAJAR

DELETE_ESPACIO

UTILIDAD LIBERAR UN ESPACIO DE MEMORIA OCUPADO A FIN DE QUE RETORNE O FORME PARTE DE LA MEMORIA Y PUEDA SER USADA SE USADO EN OTRA ACTIVIDAD

ENTRADA DIRECCION INICIAL DEL ESPACIO DE MEMORIA A SER LIBERADO

SALIDA NINGUNO

PRECONDICION LA DIRECCION DE MEMORIA A LIBERAR DEBE EXISTIR Y PERTENECER A LA DIRECCION EN USO

POSCONDICION: MEMORIA INCREMENTADO EN SU CAPACIDAD DE DIRECCIONES LIBRES PARA TRABAJAR

PONER_DATO

UTILIDAD ASIGNAR UN VALOR O UN ESPACIO CONSIDERANDO SU IDENTIFICADOR O PORTA UNA DIRECCION DE MEMORIA

ENTRADA DIRECCION INICIAL DE MEMORIA, IDENTIFICADOR DE MEMORIA, VALOR

SALIDA NINGUNA

PRECONDICION DIRECCION INICIAL DE MEMORIA E INDICADOR VALIDO

POSCONDICION MEMORIA MODIFICADA CON EL VALOR ASIGNADO A LUGAR QUE CORRESPONDE

OBTENER DATO

UTILIDAD OBTENER UN VALOR DE UN ESPACIO DE MEMORIA CONSIDERANDO SU IDENTIFICADOR Y PORTA UNA DIRECCION DE MEMORIA

ENTRADA DIRECCION INICIAL DE MEMORIA, IDENTIFICADOR DE MEMORIA

SALIDA VALOR QUE SE ENCUENTRA EN EL LUGAR ESPECIFICADO SEGUN EL INDIFICADOR

PRECONDICION DIRECCION INICIAL DE MEMORIA E INDICADOR VALIDO

POSCONDICION NINGUNA

ESPACIO OCUPADO

UTILIDAD DETERMINA CUANTOS ESPACIO DE MEMORIA ESTAN OCUPADOS

ENTRADA NINGUNA

SALIDA NUMERO DE ESPACIO DE MEMORIA OCUPADO

PRECONDICION NINGUNA

POSCONDICION NINGUNA

dir_libre

UTILIDAD VERIFICA SI UNA DIRECCION DE MEMORIA ESTA LIBRE

ENTRADA DIRECCION DE MEMORIA

SALIDA VALOR BOOLEANO / VERDADERO O FALSO.

PRECONDICION LA DIRECCION DEBE SER VALIDA

POSCONDICION NINGUNA

ESPACIO_DISPONIBLE

UTILIDAD DETERMINA CUANTOS ESPACIOS DE MEMORIA ESTAN LIBRES PARA PODER TRABAJAR

ENTRADA NINGUNA

SALIDA NUMERO DE ESPACIOS DE MEMORIA LIBRES

PRECONDICION NINGUNA

POSCONDICION NINGUNA

IMPLEMENTACION DE TDA

CONSTANTE MAX = 20
NULO = -1

DEFINIENDO TIPOS DE DATOS.

NodoM
Dato TipoDato (ENTERO)
id Cadena (12)
link Direccion de Memoria (Entero)

Fin tipo

Clase Csmemoria

Atributos
MEM arreglo (MAX) de tipo NodoM
Libre Direccion de memoria (entero)

METODO
Constructor // Crear
direccion new_espacio (cadena)
Delete_espacio (Dir)
poner_dato (dir, cadena_id, valor)
Tipo de dato obtenerDato (dir, lugar)
entero Espacio_Disponible ()
entero Espacio_Ocupado ()
booleano dir_libre (dir)

Fin de la clase

CONSTRUCTOR CSMEMORIA::CREADA
INICIO
PARA CADA i DESDE 0 HASTA max
MEM [i].LINK = i+1
FIN PARA
MEM [max].LINK = -1
LIBRE = 0
FIN

https://youtu.be/y-xP9q18phQ

CSMEMORIA::DELETE_ESPACIO (DIR)
INICIO
X = DIR
MIENTRAS MEM [x].LINK <> -1
X = MEM [x].LINK
FIN MIENTRAS
MEM[x].LINK = LIBRE
LIBRE = DIR
FIN

https://youtu.be/M6pG8zWvrNs

```
DIRECCION CSMEMORIA::NEW_ESPACIO(CADENA)
  // cadena 'a,b,c'
INICIO
    cant = NUMERO_IDS(cadena)
    DIR = LIBRE
    D = LIBRE
    PARA CADA i = 1 HASTA CANT-1
        MEM[D].ID = Obtener_Id(cadena, i)
        D = MEM[D].LINK
    FIN PARA
    LIBRE = MEM[D].LINK
    MEM[D].LINK = -1
    MEM[D].ID = Obtener_Id(cadena, cant)
  // RETORNAR DIR
FIN
```

https://youtu.be/nYtw5y_HD-60

```
NUMERO CSMEMORIA::ESPACIO_DISPONIBLE()
// Cantidad de memoria disponible
INICIO
    X = LIBRE
    C = 0  // CONTADOR
    MIENTRAS X <> -1
        C = C+1
        X = MEM[X].LINK
    FIN MIENTRAS
    // RETORNAR C
FIN
```

https://youtu.be/NIFCPg-WnTM

```
BOOLEANO CSMEMORIA::dir_libre(DIR)
// si DIR Libre Verdadero sino FALSO
INICIO
    X = LIBRE   C=FALSO  // BANDERA
    MIENTRAS (X<>-1) Y (C=FALSO)
        SI X = DIR ENTONCES C=VERDADERO
        FIN SI
        X = MEM[X].LINK
    FIN MIENTRAS
    RETORNAR C
FIN
```

https://youtu.be/GT2IvpCCWDK

```
CSMEMORIA::PONER_DATO
          (DIR, CADENA_Id, VALOR)
// FORMATO DE CADENA_id_'->nom_id'
INICIO
    Z = DIR
    ELIMINAR_Fecha(cadena_id)
    MIENTRAS Z <> NULO Z <> NULO
        SI MEM[Z].ID = cadena_id ENTONCES
            MEM[Z].DATO = VALOR
        FIN SI
        Z = MEM[Z].LINK
    FIN MIENTRAS
FIN
```

https://youtu.be/MIRYZIiiZdY

```
TIPODATO SMEMORIA :: OBTENER_DATO
                          (DIR, cadena_id)
INICIO
    Z = DIR   EX = FALSO
    Eliminar_flecha (cadena_id)
    MIENTRAS (Z <> NULO)
        SI MEM[Z].ID = Cadena_id ENTONCES
          // RETORNAR MEM[Z].DATO
        FIN SI
      Z = MEM[Z].LINK
    FIN SI
    Z = MEM[Z].LINK
  FIN MIENTRAS
FIN
```

https://youtu.be/X9GGIFIH2YA

```
NUMERO CSMEMORIA :: ESPACIO_OCUPADO
  // Cantidad de memoria ocupada

INICIO
    C = (max + 1) - Espacio_Disponible
    RETORNAR C
FIN
```

https://youtu.be/FpMxLb61210