UNIDAD I

Introducción a la programación visual #1

TIPOS POR VALOR Y REFERENCIA

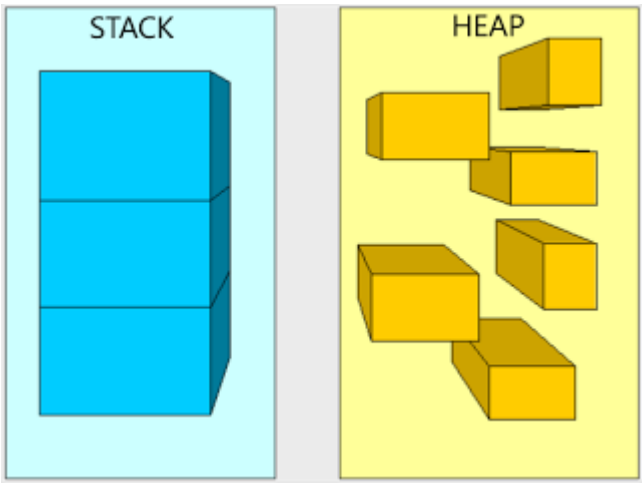
Esta clasificación es acorde a si una variable almacena su valor o es un puntero/referencia a los datos.

- **POR VALOR**:

  - Mantiene los datos dentro de su propia asignación de memoria.
  - Todos los tipos numéricos, boolean, char, date.
  - Se crea en el ***STACK***(responsable por hacer el seguimiento de la ejecución de lo que está en nuestro código) de la memoria.

- **POR REFERENCIA**:

  - Mantiene una referencia a la posición de memoria donde están los datos.
  - Object, string, arrays, class.
  - Se crea en el ***HEAP***(responsable por hacer seguimiento de la mayor parte de nuestros datos) de la memoria.



BOXING Y UNBOXING

- **BOXING:** Convertir un valor a un tipo por valor (int, double, float, etc) a un tipo por referencia (object, string, etc). Ejemplo:

  int G = 123;
  Object O = G;

- **UNBOXING:** Convertir un tipo por referencia a un tipo por valor. Ejemplo:

  Object O = 123;
  int i = (int) O;

CONVERT

Convierte un tipo de datos base en otro tipo.

- **IMPLÍCITAS**: No se requiere sintaxis especial porque la conversión es segura y no se perderán datos. Ejemplo:

  LONG ValorGrande;
  INT Entero = 5;
  ValorGrande = Entero;

- **EXPLÍCITAS**: Interviene el programador porque puede haber pérdida de datos. Ejemplo:

  DOUBLE ValorGrande = 5555.5555;
  INT Entero;
  Entero = (int) ValorGrande;

ARRAYS

- **ESTÁTICOS**
  - int[] array = new int[2]

- **DINÁMICOS**
  - array.resize(ref array, 4);

- **MATRIZ (array multidimensional)**
  - int[,] matriz = new int [100,200]

Introducción a la programación visual #2

CONTROLES DE USUARIO

Todos los controles tienen 3 características principales:

1) **COMPORTAMIENTO**: Son funciones que tienen los controles.
2) **EVENTOS**: Reacciones que tienen los controles ante distintos contextos.
3) **PROPIEDADES**: Son 'variables' que permiten definir el estado de un control.

UNIDAD II - Estructuras dinámicas I

TIPOS DE DATOS ABSTRACTOS

Un **TIPO DE DATO ABSTRACTO** (TDA) es un conjunto de datos u objetos al cual se le asocian operaciones.

TAD = VALORES + OPERACIONES

Una **ABSTRACCIÓN** es la simplificación de un objeto de la realidad en la que sólo se consideran los aspectos más relevantes.

RECURSIVIDAD

Las **FUNCIONES RECURSIVAS** (equivalentes a estructuras tipo 'bucle') son funciones que se invocan a sí mismas.

- Toda recursividad debe tener una condición de parada.
- La recursividad ocupa memoria a medida que crece. ERROR => *StackOverflow*

Listas Enlazadas

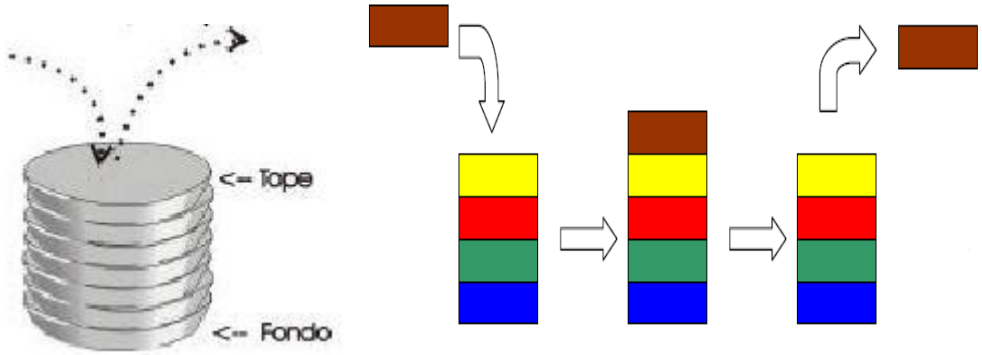
Es una estructura de datos en la cual los elementos almacenados en la misma pueden ser agregados, borrados y accedidos sin restricciones.
Se compone de **NODOS**.

OPERACIONES:

- **RECORRIDO**: Procesa cada elemento de la estructura.
- **BÚSQUEDA**: Recupera la posición de un elemento específico
- **INSERCIÓN**: Adiciona un nuevo elemento a la estructura.
- **BORRADO**: Elimina un elemento.
- **ORDENACIÓN**: Ordena los elementos de la estructura de acuerdo a sus valores.
- **MEZCLA**: Combina 2 estructuras.

1) PILAS

Una PILA(tipo LIFO) es una estructura de datos en la cual los elementos almacenados en la misma se agregan y se sacan del mismo lugar, llamado ***TOPE***(es el único lugar a partir del cual se pueden acceder a los elementos de la estructura).

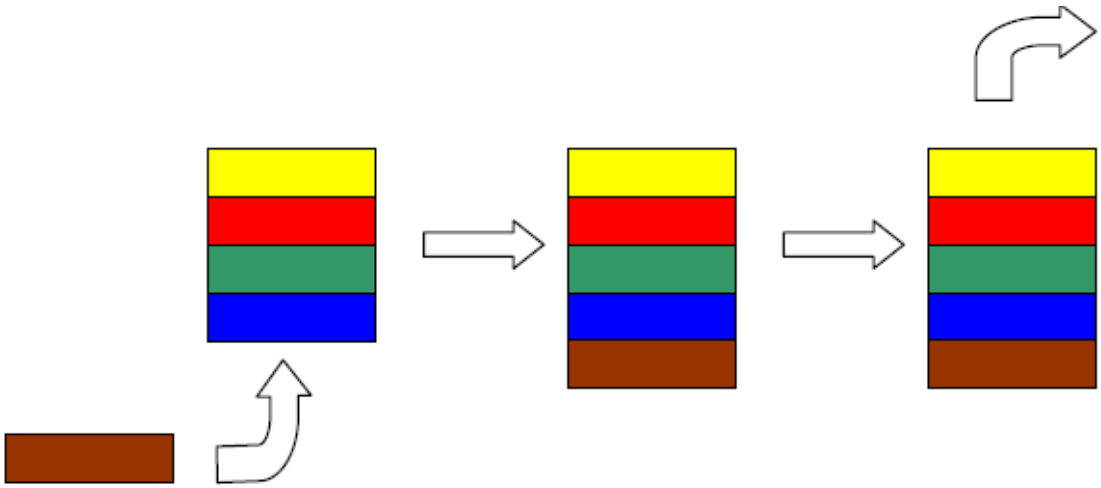


OPERACIONES

- **PUSH**
- **POP**
- **TOP/PEEK**
- **VACÍA**
- **LLENA**

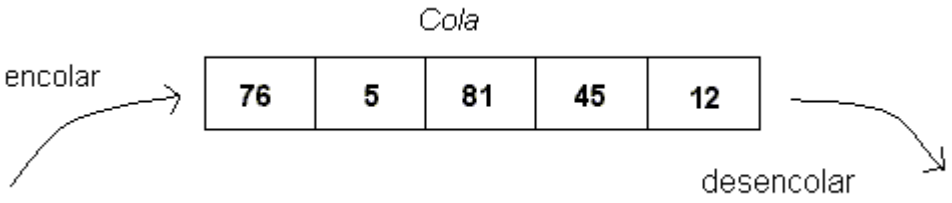
2) COLAS

Una COLA(tipo FIFO) es una estructura de datos en la cual los elementos almacenados en la misma se agregan al final y se sacan al principio de la cola.



OPERACIONES

- **ENQUEUE**
- **DEQUEUE**
- **PEEK**
- **VACÍA**
- **LLENA**



UNIDAD III - Estructuras dinámicas II

Árboles #1

Un **ÁRBOL** es una estructura de datos no secuencial, que puede definirse recursivamente como:

- una estructura vacía
- un nodo más un número finito de subárboles

ÁRBOLES AVL

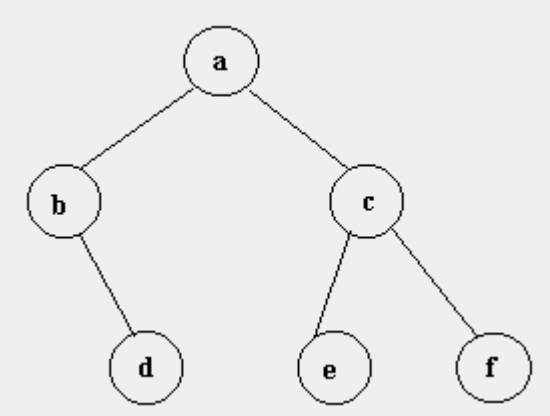
Árbol Binario Balanceado => Simplifica las operaciones de búsqueda.

- Está **BALANCEADO** cuando para todos sus nodos la altura de la rama izquierda no difiere más de una unidad de la altura derecha

Factor de Equilibrio es la diferencia entre las alturas de un subárbol con las del otro subárbol: FE = 0, -1 ó 1.

1) FE = 0 => Nodo Perfectamente Equilibrado
2) FE = 1 => Nodo derecho es un nivel más alto
3) FE = -1 => Nodo izquierdo es un nivel más alto
4) FE >=|2| => Hay que equilibrar el árbol.

NOMENCLATURA



1) **RAÍZ** => Elemento que no tiene antecesor. Ejemplo: A
2) **RAMA** => Arista entre 2 nodos.
3) **ANTECESOR** => Un nodo X es antecesor de un nodo Y y si por laguna de las ramas Y se puede llegar a X
4) **SUCESOR** => Un nodo X es sucesor de un nodo Y si por alguna de las ramas de Y se puede llegar a X
5) **GRADO DE UN NODO** => Número de descendientes directos que tiene. Ejemplo: C tiene grado 2.
6) **HOJA** => Nodo que no tiene descendientes: grado 0. Ejemplo: D.
7) **NODO INTERNO** => Aquel que tiene al menos un descendiente
8) **NIVEL** => Número de ramas que hay que recorrer de la raíz a un nodo. Ejemplo: Nivel nodo A = 1; Nivel nodo E = 3.
9) **ANCHURA** => Es el mayor valor del número de nodos que hay en un nivel.

RECORRIDO DE UN ÁRBOL

1) RECORRIDO EN PROFUNDIDAD

- **PREORDEN**: (raíz, izquierdo, derecho). Para recorrer un árbol binario no vacío en preorden, hay que realizar las siguientes operaciones recursivamente en cada nodo, comenzando con el nodo raíz:

    - Visite la raíz.
    - Atraviese el sub-árbol izquierdo.
    - Atraviese el sub-árbol derecho.

- **INORDEN**: (izquierdo, raíz, derecho). Para recorrer un árbol binario no vacío en inorden hay que realizar las siguientes operaciones recursivamente en cada nodo:

    - Atraviese el sub-árbol izquierdo.
    - Visite la raíz..
    - Atraviese el sub-árbol derecho.

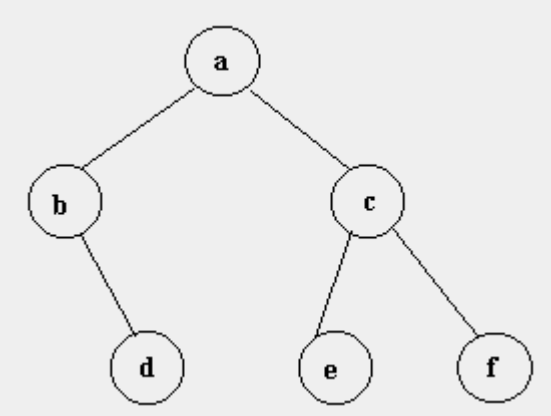
- **POSTORDEN**: (izquierdo, derecho, raíz) hay que realizar las siguientes operaciones recursivamente en cada nodo:

    - Atraviese el sub-árbol izquierdo.
    - Atraviese el sub-árbol derecho.
    - Atraviese la raíz.

2) RECORRIDO EN AMPLITUD

Consiste en ir visitando el árbol por niveles. Primero se visitan los nodos del nivel I (la raíz), después los nodos del nivel II, así hasta que ya no queden más.Si se hace el recorrido en amplitud del siguiente árbol se visitará los nodos en el orden: a,b,c,d,e,f

El recorrido se hará de forma iterativa utilizando una cola como estructura de datos auxiliar.



ÁRBOL BINARIO DE BÚSQUEDA

Un árbol binario de búsqueda es aquel que es:

-Una estructura vacía
ó
-Un elemento o clave de información (nodo) más un número finito (a lo sumo dos) de estructuras tipo árbol, disjuntos, llamados subárboles y además cumplen lo siguiente:

- Todas las claves del subárbol izquierdo al nodo son menores que la clave del nodo.
- Todas las claves del subárbol derecho al nodo son mayores que la clave del nodo.
- Ambos subárboles son árboles binarios de búsqueda.

UNIDAD IV - ARCHIVOS

ARCHIVO => Conjunto de bits que son almacenados en un dispositivo.

- Se almacenan en DISCOS.
- Es identificado por un nombre y la descripción de la carpeta.

Ventajas:

1. Residencia
2. Independencia
3. Permanencia
4. Portabilidad
5. Capacidad

Desventajas:

1. Concurrencia
2. Transacciones
3. Dependencia con el programa que lo gestiona

TIPOS DE ARCHIVO

1) **ARCHIVOS PERMANENTES**: Los datos guardados en ellos cambian poco en el tiempo. Son generalmente usados para estadísticas y cumplen una función de soporte.

2) **ARCHIVOS DE MOVIMIENTO**: Son los tradicionales archivos donde el sistema realiza las actuaciones constantes.

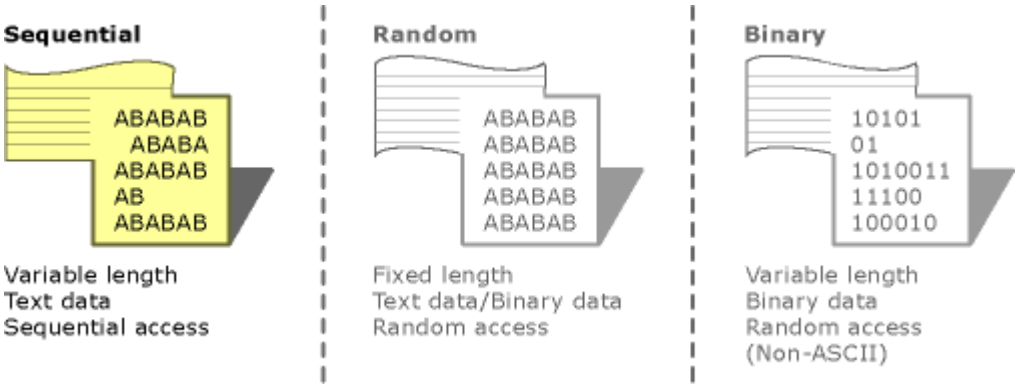
3) **ARCHIVOS TEMPORALES**: Se crea solo cuando se usa y luego se destruye.

Los ARCHIVO DE TEXTO PLANO son aquellos que están compuestos únicamente por texto sin formato, sólo caracteres.

Un ARCHIVO BINARIO es un archivo informático que contiene información de cualquier tipo, codificada en forma *binaria* para el propósito de almacenamiento y procesamiento de ordenadores.

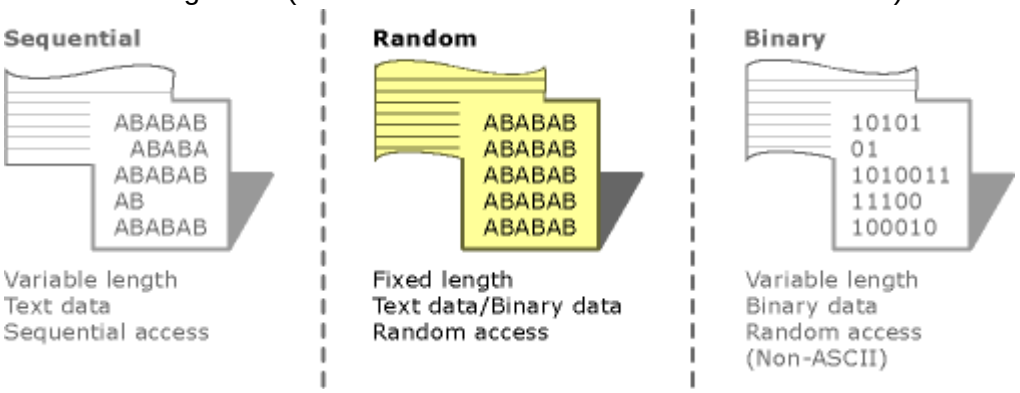
ACCESO SECUENCIAL

- Diseñado para utilizarlo con archivos de texto sin formato.
- Los datos se almacenan como caracteres **ANSI**



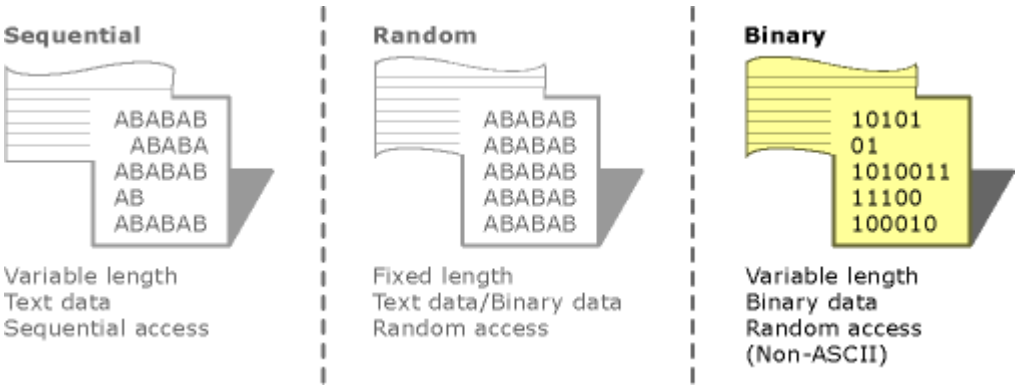
ACCESO ALEATORIO

- Compuesto por un conjunto de campos que contienen información
- Se pueden utilizar tipos definidos por el usuario para crear registros compuestos por numerosos registros. (datos se almacenan como información binaria)



ACCESO BINARIO

- Permite utilizar archivos para almacenar datos de cualquier modo que se ajuste a las necesidades, ya sean numéricos, de cadena o ambos.
- Es necesario conocer cómo se escribieron exactamente los datos en el archivo para poder recuperarlos correctamente.
- **Ejemplo:** Si almacena una serie de nombres y números de teléfono, debe recordar que el primer campo (el nombre) es texto y el segundo (el número de teléfono) es numérico.



CAMPOS DE UN REGISTRO

- Son la unidad lógica donde se almacenan los datos de un registro.
- Tiene como característica nombre, tipo y tamaño.
- Conjunto de campos = **REGISTRO**
- Conjunto de registros = **ARCHIVO**

NOMBRE	TIPO	TAMAÑO
NroCta	Numero	5
RazonSocial	Alfabetico	25
FechaDeposito	Fecha	6
Comentarios	Memo	200

SYSTEM.IO

Proporciona métodos estáticos para crear, copiar, eliminar, mover y abrir un solo archivos y contribuye a la creación de objetos **FileStream.**

Clase	Descripción
FileStream	Permite crear objetos para leer o escribir datos en archivos. Para ello es necesario definir el nombre completo del archivo (incluyendo la ruta de ubicación de sus carpetas)
StreamWriter	Permite crear objetos que implementan un sistema de escritura de datos basado en secuencias de caracteres.
StreamReader	Permite crear objetos que implementan un sistema de lectura de datos basado en secuencias de caracteres
File	Esta clase contiene métodos estáticos para manipular archivos como su creación, copiarlos, eliminarlos, moverlos o detectar su existencia.

FILEMODE

FileMode	Uso
CreateNew	Crea un nuevo archivo. Si el archivo existe dispara una IOException
Truncate	Abrir un archivo existente. Una vez abierto, el archivo será truncado a cero bytes de longitud.
Create	Crea un nuevo archivo. Si el archivo existe será sobrescrito.
Open	Abrir un archivo existente. Si no existe dispara una FileNotFoundException.
OpenOrCreate	Abrir un archivo existente, si no existe, lo crea
Append	Abrir un archivo para agregar datos al final en caso de existir; de lo contrario crea un archivo nuevo

FILEACCESS

FileAccess	Uso
Read	Acceso al archivo en modo de solo lectura
ReadWrite	Acceso al archivo en modo de lectura y escritura
Write	Acceso al archivo en modo de solo escritura

CORTE DE CONTROL

- Es una forma ordenada de mostrar información jerárquicamente.
- **USO**: aplicación más común: 'generar reportes que acumulen cantidades y/o importes.'

CAMPO DE CONTROL

- Es el campo que identifica a cada grupo de elementos de entrada de un conjunto mayor de datos.

SI EL ARCHIVO NO ESTÁ ORDENADO, NO SE PUEDE APLICAR CORTE DE CONTROL

APAREO DE ARCHIVOS

- Se realiza con 2 archivos, que estén ordenados por el mismo campo.
- El corte de control del proceso será hasta que ambos o alguno de los archivos finalice.

RESUMEN

CORTE DE CONTROL: Es una técnica de procesamiento de datos ordenados por diversos criterios, que permite agruparlos para obtener subtotales.

APAREO: Es una técnica de procesamiento que involucra 2 archivos con datos ordenados y permite generar una salida combinada a partir de estos 2 archivos.

UNIDAD VI - ESTILOS DE PROGRAMACIÓN

¿Qué es un paradigma de programación?

Representa una particular manera de enfocar los procesos de construcción de software.

Diferentes formas de pensar la **solución de problemas** y en diferentes **estilos de programación**.

Todos los paradigmas tienen en común el uso de los siguientes métodos:

1) **MODULARIDAD**: Descomposición lógica de un sistema en entidades más pequeñas.

2) **RECURSIVIDAD**: Significa aplicar una función como parte de la definición de esa misma. Ejemplo: cálculo del factorial