**Estructura básica de datos:**

1. Salida de datos por pantalla **(Escritura).**
2. Entrada de datos por el usuario **(Lectura).**
3. Operaciones matemáticas.
4. Declaración y uso de variables.
5. Tipos de datos elementales.
6. **SALIDA BÁSICA POR PANTALLA:**
   1. **System.out.println** 🡪 salto de línea
   2. **System.out.print**
   3. **System.out.printf** 🡪 Marcas de formato
      1. %d para datos enteros
         1. %5d o %05d mostrar al menos5 cifras, se rellena con espacios a la izquierda o ceros a la izquierda.
      2. %f para datos reales
         1. %5.2f mostrar 5 elementos (entero, punto coma y decimal) de los cuales 2 son decimales. Espacios hacia la izquierda, ceros a la derecha.
      3. %s para cadenas de texto
      4. %c para caracteres

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Se pueden enlazar (**concatenar**) con el operador **+.**

1. **ENTRADA DE DATOS POR EL USUARIO:**
   1. **Scanner**: permite procesar tipo de datos simples de una fuente de entrada.
   2. **System.in.**
   3. Incluir **import .**
   4. Instrucciones:
      1. **nextLine()** leer texto (hasta que se pulse intro.
      2. **nextInt()** leer un entero.
      3. **nextFloat()** leer un float.
      4. **nextBoolean()** leer un boolean.
   5. **System.console().readLine()**
2. **OPERADORES**: Son elementos que permiten realizar una operación determinada en un programa.
   1. **Aritméticos**: Operaciones arimeticas.
      1. **Suma (+) , Resta (-) , Multiplicación (\*) , División (/) y Resto de división (%)**
      2. Precedencia:
         1. \* , / , % de izquierda a derecha
         2. + , - de izquierda a derecha
         3. El uso de paréntesis permite variar precedencia.
   2. **Asignación**: asignar valor o modificarlo.
      1. **Asignación simple (=) , Autosuma (+=) , Autorresta (-=) , Automultiplicación (\*=) , Autodivisión (/=), Autorresto (%=)**
   3. **Relacionales:** 
      1. **Mayor que (>) , Mayor o igual que (>=) , menor que (< ) , menor o igual que (<=), igual que (==) , Distino de (!=)**
      2. **.equals()**
   4. **Lógicos:** 
      1. **Y (&&) , O (||) , No (!) (TABLAS DE VERDAD)**
      2. Precedencia:
         1. Paréntesis
         2. Negaciones
         3. &&
         4. ||

**\*Comprobación cortocicuito:** Si la primera comprobación es falsa (&&) la comprobación es falsa, en un O si la primera comprobación es verdadera (||) la comprobacióin es verdadera

1. **DECLARACIÓN DE VARIABLES**: Espacio de memoria donde podemos almacenar un valor.
   1. **tipoDato nombreVariable = valor;**
2. **TIPOS DE DATOS:**
   1. Tipos de datos enteros
      1. **byte (1) , short (2) , int (4) , long(8)**
   2. Tipos de datos reales:
      1. **float (6-7 cifras) , double (15 cifras)**

**Desbordamiento:** hacer que salga más allá de los limites el valor de un dato. Esto tiene como consecuencia 🡪 Valores no esperados / errores de programa

* 1. Tipo de dato Carácter:
     1. **Char:** Permite representar cualquier carácter o símbolos sueltos (letras, dígitos, puntuación, tabulaciones, saltos de línea..)
     2. Se representan entre ‘ ‘.
     3. Secuencias de escape: ‘\n’ 🡪 Salto de línea| ‘\t’🡪 tabulación| ‘\”’🡪 mostrar comillas dobles por pantalla| ‘\’’🡪 comillas simple por pantalla| ‘\\’🡪 barra invertida por pantalla
  2. Tipo de datos Cadena de caracteres:
     1. **String:** representa una secuencia de caracteres.
     2. Se representa entre “ ”.
     3. Secuencias de escape como en char.

/\*

\*Autoincremento: n++, ++n

\*Autodecremento: n-- , --n

\*/

Interfaz de usuario gráfica

Descripción generada automáticamente con confianza baja

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

**Programación estructurada:**

1. Condiciones**: Nos permite elegir entre varios caminos o alternativas posibles, en función de una condición o condiciones que deberán cumplirse para uno u otro.**
2. Bucles: Permitirán repetir un conjunto de instrucciones un número determinada de veces, o mientras se cumpla una determinada condición.
3. **TIPO DE DATO BOOLEANO:**
   1. Solo puede tomar dos valores: True o False.
   2. Se representan por la palabra boolean.
4. **ESTRUCTURA IF:**
   1. La estructura evalúa una expresión booleana entre paréntesis y ejecuta una instrucción o conjunto de instrucciones si dicha expresión es cierta.
   2. if-else – Si la condición evaluada no es cierta y no se ejecuta la instrucción if podemos usar else para ejecutar la segunda instrucción.
   3. Else – if - ejecución de varios caminos.

Interfaz de usuario gráfica

Descripción generada automáticamente

1. **ESTRUCTURA SWITCH:**
   1. Permite evaluar el valor de una variable o expresión y ejecutar un conjunto de instrucciones u otro en función de dicho valor.
   2. Se coloca la variable o expresión a evaluar dentro del paréntesis de la partícula Switch y se define un bloque case por los posibles valores que contemplamos.
   3. Hay que poner break sino al ejecutar un case se ejecutaran el resto de los siguientes. Es decir, para salir del bloque switch en general.
   4. Default para dar respuesta en el caso de que ninguno de los cados de la expresión coincida con los examinados.

Imagen que contiene Interfaz de usuario gráfica

Descripción generada automáticamente

1. **OPERADOR TERNARIO:**

Imagen que contiene Interfaz de usuario gráfica

Descripción generada automáticamente

1. **ESTRUCUTURA REPETITIVA WHILE:**
   1. Permite ejecuta una instrucción o instrucciones que engloba mientras se cumpla la condición expresada en paréntesis.

Interfaz de usuario gráfica, Texto

Descripción generada automáticamenteTexto

Descripción generada automáticamente

1. **ESTRUCUTURA REPETITIVA DO-WHILE:**
   1. Permite ejecutar la instrucción o instrucciones que engloba mientras se cumpla la condición expresada en paréntesis.
   2. Diferencia con While la condición se evalúa después de ejecutar al menos una vez el conjunto de instrucciones. Lo que permite la ejecución previa a la evaluación.

Interfaz de usuario gráfica, Texto

Descripción generada automáticamenteInterfaz de usuario gráfica, Texto

Descripción generada automáticamente

1. **ESTRUCUTURA REPETITIVA FOR:**
   1. Permite ejecutar la instrucción o instrucciones que engloba un número determinado de veces, mientras se cumpla la condición expresada entre paréntesis.
   2. Se diferencia de la estructura while o do-while en que la variable que se incrementa y hace de contador de repeticiones se integra y modifica en la propia estructura for.

Gráfico

Descripción generada automáticamente con confianza media

Gráfico de líneas

Descripción generada automáticamente

1. **ESTRUCUTURA REPETITIVA BREAK Y CONTINUE:**
   1. BREAK: Permite salir de la estructura repetitiva donde se esté de forma inmediata
      1. Se suele hacer cuando dentro de una repetición “normal” se d una condición excepcional que obligue a interrumpir las iteraciones.
      2. No recomendable.
   2. CONTINUE: fuerza una repetición más de la estructura iterativa donde se esté, de forma inmediata.
      1. Suele hacerse cuando dentro de una repetición “normal”, se da una condición excepcional que obligue a dejar de ejecutar la instrucción restante y pasar a la siguiente repetición del bucle.
      2. No recomendable.

**Texto

Descripción generada automáticamente**Imagen que contiene Interfaz de usuario gráfica

Descripción generada automáticamente

1. **RECOMENDACIONES DE USO:**
   1. While:
      1. Numero de repeticiones indefinidas.
      2. Ni siquiera se sabe si habrá una primera repetición.
      3. EJ*: Programa que, dado un número, determine cuantes veces seguidas es divisible por dos.*
   2. Do..While:
      1. Numero de repeticiones indefinidas.
      2. Al menos se sabe que habrá una repetición.
      3. EJ: *Programa que le pida al usuario números hasta que este escriba un 0.*
   3. While:
      1. Numero de repeticiones conocido e invariable antes de empezar el bucle
      2. EJ: *Programa que cuente del 1 al 10.*
2. **BUCLES ANIDADOS:**
   1. Se pueden incluir bucles dentro de bucles, tantos niveles como se necesiten.
   2. Esto permite repetir el bucle tantes veces como indique el exterior.