

Introducción a la Programación

Algoritmo: Conjunto ordenado y finito de operaciones que permiten hallar la solución de un problema

Características:

Preciso, Definido y finito.

Conceptos:

Algoritmo, Pseudocódigo / Diagrama, Programa fuente, Lenguaje de programación, Código Máquina y Compilador

Pseudocódigo: Código que escriben las personas para imaginarse las órdenes que le darán a la computadora a través de un lenguaje de programación.

Para diseñar algoritmos hay que:

- * Entender lo que te pide
- * Visualizar cuáles son los datos de entrada y salida
- * Poner en orden los pasos
- * Ser específico en los pasos
- * Pensar en los inconvenientes
- * Tratar de resolverlo en el menor número de pasos
- * Probarlo varias veces analizando los resultados.

El sistema de computo se compone en:

Hardware

- Se puede tocar, tendrá un nombre y forma parte del equipo de computo

Software

Controla el hardware

Se divide en Software de aplicación y Sistema

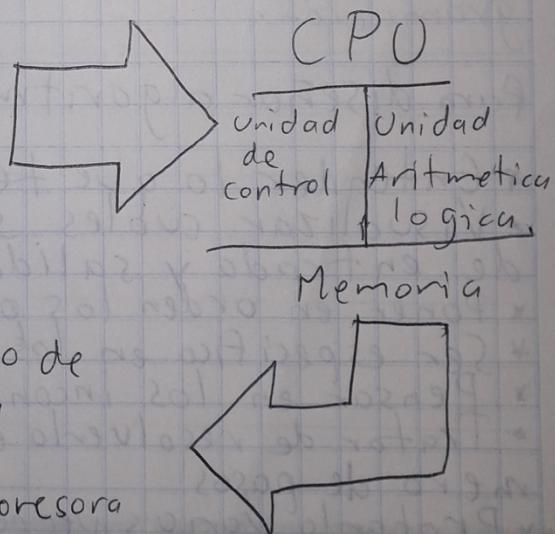
Organización física de la computadora

Dispositivos de entrada

Teclado, ratón, etc.

Dispositivo de salida

Monitor, impresora etc.



• ¿Qué es programación?
R= Es programar

• ¿Qué es programar?
R= elaborar programas

• ¿Qué es un programa?
R= Operaciones, que en un orden determinado, ejecutan ciertas máquinas.

Procesamiento de información en la computadora

Datos de entrada → Proceso → Información de salida

Si es entrada es dato y si es salida es información

Los algoritmos se expresan en pseudocódigo y diagramas de flujo

Pseudocódigo

- No lo puede ejecutar la computadora
- Concentrarse en la lógica
- Fácil de traducir a lenguaje de programación.

Ejemplo: este espacio se llama:
Indentación

- 1 Proceso Saludo
- 2 Definir nombre como carácter
- 3 Escribir '¿Cómo te llamas?' ;
- 4 Leer nombre
- 5 Escribir 'Hola ', nombre;
- 6 FinProceso

La sintaxis de los lenguajes de programación es más sencilla que aprender otro idioma como inglés, francés, etc.

Al final de cada instrucción pon punto y coma (;) en los lenguajes de programación formales por que es obligatorio

En pseudocódigo:

Letras azules: Palabras reservadas con significado especial en pseudocódigo.

Letras rojas: Texto tal cual, aparece cuando pones comillas.

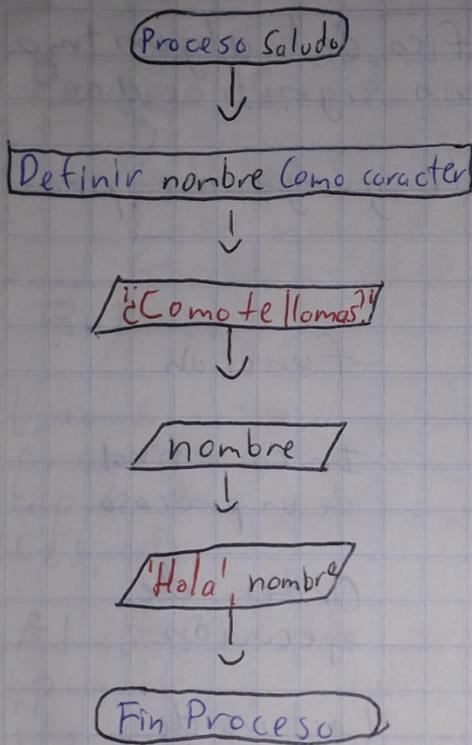
Letras negras: identificadores

Diagramas de flujo

- Representación gráfica del algoritmo
- Utilizan simblos con significados bien definidos.
- Fácil de traducir a lenguaje de programación formal

Simbolo	Nombre	función
	Inicio/ Final	Inicio y final de un proceso
	Línea de Flujo	Orden de ejecución
	Entrada/ Salida	Lectura de datos E/S
	Proceso	Cualquier tipo de operación
	Decisión	Analizar situación

Ejemplo:



Ejercicios Tarea

1-

- 1 - Ver en el carro cual es la llanta ponchada
- 2 - Ver especificaciones de la llanta
- 3 - Ir a un lugar donde vendan la llanta
- 4 - Comprar la llanta
- 5 - Volver a donde está el carro con la llanta ponchada
- 6 - Desatornillar los tornillos de la llanta ponchada
- 7 - Quitar la llanta ponchada

- 8 - Colocar la llanta nueva
- 9 - Atornillar la llanta nueva al carro
- 10 - Manejar el carro con la llanta nueva para ver si no hay problemas.

- 1 - Comprar los ingredientes
- 2 - Lavar los recipientes y utensilios que voy a utilizar
- 3 - Tener todo a mano
- 4 - Precalentar el horno a 300°
- 5 - Mezclar los siguientes ingredientes: 60 g de levadura y 15 ml de leche hasta tener una masa
- 6 - En un molde prestando atención a la masa y estirarla hasta cubrir toda el area del molde
- 7 - En cima de la masa, poner salsa de tomate hasta cubrir la masa
- 8 - Una vez puesto la salsa de tomate, cubrir la masa con queso
- 9 - Meter al horno y dejar hornear de 15 a 20 min.
- 10 - Dividir en porciones.

Lenguaje de programación

Conjunto de símbolos y reglas sintácticas que permiten escribir un programa.

- Define sintaxis
- Traduce el programa a un código binario (código objeto)

El pseudocódigo o diagramas de flujo no son comprensibles por la computadora se deben codificar

El algoritmo escrito en un lenguaje de programación se denomina código fuente.

Cada lenguaje tiene:

- * Instrucciones en entrada/salida
- * Instrucciones de cálculo
- * Instrucciones de control

Lenguajes de bajo nivel

Van muy relacionados con la máquina y el sistema operativo

Lenguaje Máquina

0101
1001
0101
1110 → Computadora
Hardware

Ensamblador

ADD R1 F4
Mov F4 C2 → Ensamblador → 0101
1101
0101

Lenguaje Alto nivel

Están cercanos al lenguaje natural

Comprendida más fácil.

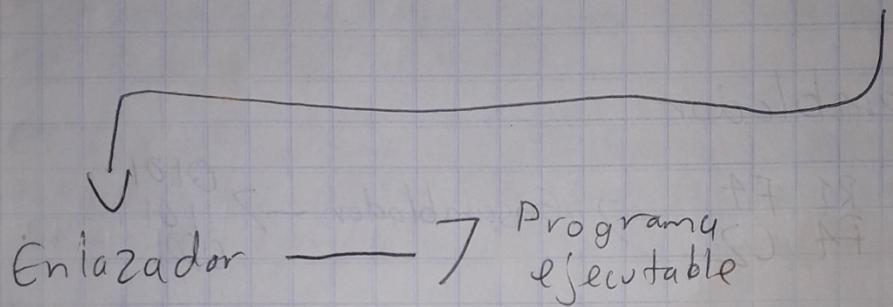
Traductores

Compiladores: Compilan y enlazan programas completos.

Interpretes: Compilan y ejecutan instrucciones a instrucción

Fase de compilación

(Código fuente → Compilador → Código objeto
(Código Java C#))
(011010110)



Paradigmas de programación

Imperativo

→ estructurada

→ Modular

→ Orientada a objetos

Declarativo

- └→ Lógica
- └→ funcional

Programación estructurada

Secuencia de instrucciones

Como debe realizar el cálculo

- Se basa en dar órdenes
 $(A = B + C)$

Programación Modular

Dividir en bloques

Se controla con secuencia, selección e iteración

Reutilización de código.

Programación orientada a Objetos

- Abstracción de datos
PB el paradigma más usado.

Encapsular estado y operaciones de
objetos

Comunicación a través de mensajes

No existe el mejor Paradigma, el
mejor depende para qué lo quieras u-
so

- o - o - o - o - o - o -

Elementos de un programa

¿Qué debo saber para empezar a elab-
orar programas?

Elementos!

• Instrucciones

• Datos

• Variables y constantes

• Expresiones y operadores

• Estructura de control secuencial.

Instucción

Conjunto de datos insertados en una secuencia específica que el procesador interpreta y ejecuta.

Dichas instrucciones están formadas de:

- * Palabras reservadas de cada lenguaje
- * Datos
- * Operaciones
- * Expresiones

Datos

- * Expresión general que describe los objetos con los que opera la computadora.
- * Las computadoras trabajan con varios tipos de datos.
- * En las computadoras, los datos deben ser de un tipo específico.

Tipos de datos

Determinan como se representa en la computadora

Dato	Representación	Operaciones
5	Entero	Operaciones Aritméticas
"5"	Cadena de Caracter	Extraer longitud

- Datos básicos : incluidos dentro del lenguaje
- Compuestos: Construidos a partir de los básicos.
ej: Estructuras, vectores, matrices/ tablas, cadenas.
- En C o C++: punteros

Datos numéricos

- Bool = para true o false
 - Double = para decimal
 - Char
 - Signed char } para caracteres.
 - Unsigned char }
 - short int
 - unsigned short int }
 - Int } para números enteros
 - Unsigned int
 - long int
 - unsigned long int }
 - long long int }
 - unsigned long long int }
 - Float = para decimal
- En javascript para poner un número ponemos .number

Bit \rightarrow 0, 1

Byte \rightarrow 8 bits

Cuando usas un dato numérico para ingresar números asegúrate de cuantos bits es, porque si introduces un número pequeño a un dato numérico de muchos bits, la memoria Ram se va a gastar o llenar mucho.

Datos numéricos:

Entero:

- Se pueden representar con 8, 16, 32 y 64 bits

- Subconjunto finito de números enteros

Reales:

- Cualquier número, entero, con decimal y negativo y positivo

- Para números grandes ponen notaciones científicas.

3650000000...0000... $\rightarrow 3.675201 \times 10^{25}$

Tipos

Float 1×10^{-6}

Double 1×10^{301}

Datos lógicos / Booleanos (bool)

- Solo pueden tomar 2 valores: true/false.
- Se utiliza para realizar alternativas (si/no) a determinadas condiciones.

Dato carácter (char)

- Conjunto finito y ordenado de caracteres que la computadora conoce
- Tiene un solo carácter
- Los caracteres no son estándares.
 - Alfabéticos: (A-Z) (a-z)
 - Numéricos: (1, 2, ..., 9, 0)
 - Especiales: (+, -, *, ., /)

Dato cadena (string)

- Sucesión de caracteres que se encuentran delimitados por comillas simples o dobles.

- La longitud es el número de caracteres de la cadena.

- 'Hola mundo'
- "Jack Sparrow"
- '(228) 8-40-35-9t'

A JavaScript no le importa si las comillas simples o dobles para poner datos cadena.

Variables y constantes

Constante: Dato que permanece sin cambios durante el desarrollo del algoritmo.

ej: 7, 3.5, "hola", 'm'

Variable: Que puede cambiar

ej: a, x2, res, total, sueldo

Características: tienen un tipo y sólo pueden tomar valores de tipo

El nombre que se le otorga a una variable o constante se llama identificación.

Reglas

- Debe empezar con letra
- No deben existir espacios en blanco intermedios.
- La longitud depende del lenguaje
- NO palabras reservadas (if, while, int)

Recomendaciones

- Los nombres deben ser significativos y tener relación con su contenido

$$a = (b + h)/2$$

$$\text{pato} = [\text{algo1} \text{ "mesa"}]/2$$

- Usar un estandar de nomenclatura (camel case)

area Cuadrado
sum Valores

Expresiones y operadores

Expresión

- Combinación de constantes, variables o funciones, las cuales son interpretadas de acuerdo a las normas particulares de precedencia y asociación para un lenguaje de programación
- Combina valores y operaciones y generan un valor
- Se resuelven de izquierda a derecha, pero depende la jerarquía de sus operadores

Tipos de expresiones

- Simples: Dan valor numérico directamente.
- Compuestas: Se resuelven cosas y asignan variables

Operadores

- Aritméticos: ($+$, $*$, $-$, $/$, $\%$) dan números
- Relacionales: ($<$, $>$, $=$, \neq , \geq , \leq) dan V o F
- Lógicos: $V \rightarrow F$, $\neg V$, $V \& V$, $V \vee V$, $\neg V \& V$, $\neg V \vee V$, $V \neg V$, $\neg V \neg V$

$$\begin{array}{r} 1 \\ 3 \sqrt{4} \\ - \end{array}$$

1 ← esto es el %

$$\begin{array}{r} 1 \\ 3 \sqrt{5} \\ - \end{array}$$

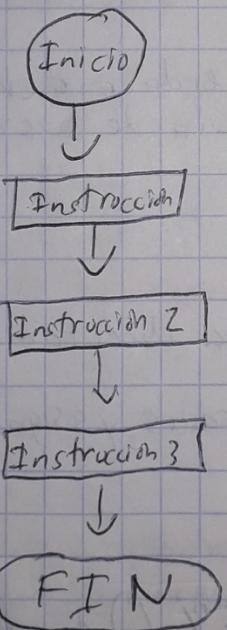
2 ← esto es el %

- 0 - 0 - 0 - 0 - 0 - 0 - 0 - 0 -

Estructura de control secuencial

Secuencia

- Aquella en la que una instrucción o acción sigue otra



en pseudocódigo:

principal ()

 inicio

 entero a, b, suma

 imprimir mensaje "introduce a"

 leer a

 imprimir mensaje "introduce b"

 leer b

 suma = a+b

 imprimir mensaje "Resultado"

 fin

Pseint

Escribir: Le dices a la computadora que lea algo: Escribir "Hola Mundo";

Leer: Permite ingresar información desde el teclado: Leer numero 1;

Asignación: Permite almacenar un valor en una variable: número 1 = 4;

Borrar pantalla: permite borrar la pantalla ~~██████████~~, se escribe "Limpiar Pantalla"

Esperar tecla: Detiene el algoritmo cuando presionas una tecla cualquiera

Cuando te sale un código de error: ej: Error 211, tienes que buscar en google que significa ese error en el programa que te saltó el error.

Para abrir para sacar capturas de pantallas en windows, teclean windows, shift, s para abrir el apartado para sacar capturas de pantalla

Metodologías del desarrollo de Software.

Modelo cascada
Etapas:

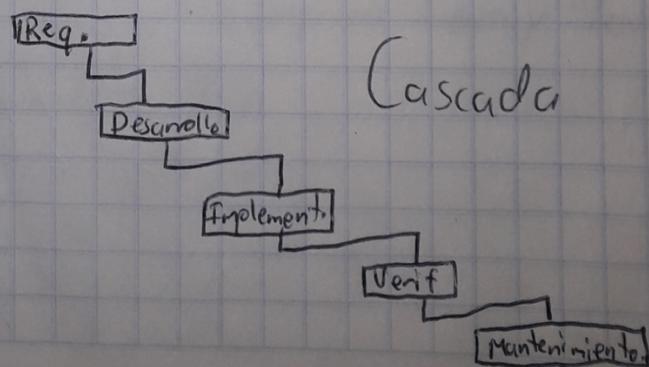
- Análisis
- Diseño en algoritmo
- Codificación
- Compilación y ejecución
- Verificación
- Depuración
- Mantenimiento
- Documentación

No es eficiente esto; el modelo cascada no funciona.

Metodologías tradicionales

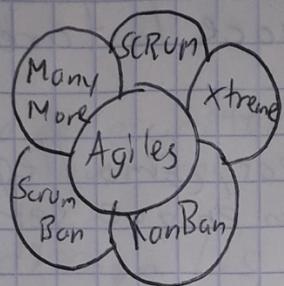


Espiral



Cascada

Metodologías Agiles



Metodología SCRUM
es la más usada

No existe metodología mejor, cuando seamos profesionistas, debemos de saber cuál aplicar y como aplicar.

Análisis

Se determina qué hace el programa

Se intenta precisar el problema

Se necesita la especificación de los requisitos del cliente o el encargado

Ejemplo:

Suma de 2
números

Entrada: dos números
Salida: la suma de dichos
números

Restricciones: Ninguno
Soluciones: Leer 2 números,
sumar los dos números y
guardar en una variable,
desplegar la variable

Diseño

- Se determina cómo se hace la tarea solicitada
- Se identifican las tareas, cómo llegar a su solución y el orden en que serán realizadas
- Divide y vencerás
- Se presenta la solución usando alguna herramienta de diseño de algoritmos.

Ejemplo:

Introducir desde el teclado las calificaciones de 3 materias, calcular y mostrar la calificación promedio.

Tareas

- Pedir calif.
- Sumar calif.
- Dividir la suma entre 3 para obtener el promedio
- Mostrar el promedio

Algoritmo

Inicio

Definir Suma Calificaciones, Promedio (Como Entrada);
Suma Calif = 0;
Escribir "escribe las 3 calif.";
Leer calif1;
Leer calif2;
Leer calif3;
Suma Calificaciones = calif1 + calif2 + calif3;

Promedio = Suma (calificaciones) / 3;
Escribir Promedio

FIN

Prueba de escritorio: ejecución manual
del algoritmo

- Se diseñan los datos de prueba que abarquen todos los caminos. Se comprueba el resultado.

No existe un estandar para escribir
y diseñar una prueba de escritorio

definir

escribir

Variable Valores Pantalla

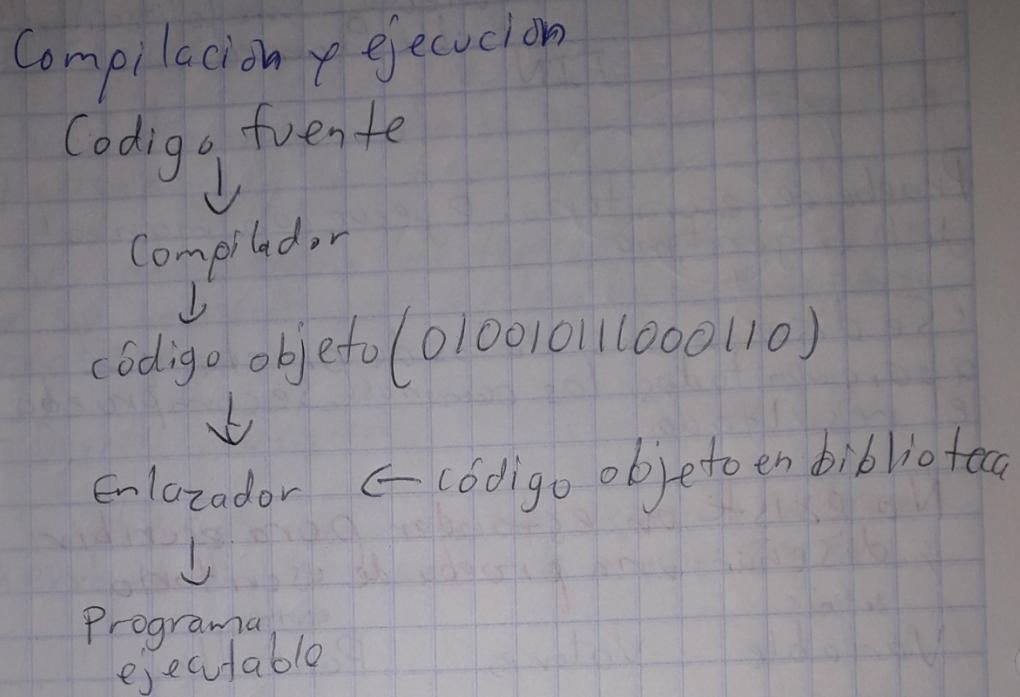
suma(calificaciones)	0,24	"Escribe 3 calificaciones"
calificación 1	7	7
calificación 2	8	8
calificación 3	9	9
Promedio	8	8

Codificación

- Escritura en un lenguaje de programación del algoritmo desarrollado en las etapas anteriores

- Conversión de las palabras en español por las palabras reservadas del lenguaje

- Código fuente.



No todos los errores los detecta el compilador

Verificación y depuración

Verificación: Ejecutar el programa con una amplia variedad de datos de entrada para determinar si hay errores.

Depuración: El proceso de encontrar los errores del programa.

Documentación y mantenimiento

- Descripción de los pasos a dar en el proceso de resolución de un problema
- Programas pobremente documentados son difíciles de leer, más difíciles de depurar y casi imposible de mantener y modificar
- Documentación interna: líneas de comentarios
- Documentación externa: análisis, diseño, manuales de usuario etc.

Para hacer comentarios en los lenguajes de programación (Algunos) se usa:

// comentarios en esa misma linea

/* */ comentarios en multilíneas

/** */ para documentación

Vital para cuando se desea corregir posibles errores o cambiar el programa.

- Después de cada cambio, la documentación se debe actualizar
- Es práctica frecuente numerar las diversas numeraciones del programa (1.0, 1.1, 2.0, 2.1)

* Si los cambios son importantes se varía el primer dígito (1.0 - 2.0), en caso de pequeños cambios se varía el segundo dígito (2.0 - 2.1)

Estructura de control selección

- Se utilizan para tomar decisiones lógicas

- Evalúan una condición y se realiza una opción u otra dependiendo del resultado.

- Las condiciones son expresiones lógicas

Existen 3 tipos

* Simples → expresiones lógicas

» Dobles

» Múltiples - evalúan 1 valor nadamas

Simples

Si (If)

Ejecutan acciones solamente cuando se cumple una condición

Si la condición es falsa, no se hace nada.

Pseudocódigo

Si <condición> entonces
<acciones>;
FinSi

C++

```
if (<condición>){  
    <acciones>;  
}
```

Dobles

Si - entonces - sino

Ejecutan unas acciones cuando la condición es verdadera y otras si la condición es falsa

Pseudocódigo

C++

```
Si <condición> entonces    i f(condición) {  
    <acción(es)>           <acción(es)>  
Sino                      } else {  
    <acciones>             <acción(es)>  
Fin Si                     }
```

Múltiples

Es muy útil cuando se necesitan más de 2 elecciones

Evaluá una expresión y dependiendo del valor elegido ejecuta sus acciones.

Pseudocódigo

Según <expresión> hacer
valor1:

<acción(es)>

valor2:

<acción(es)>

De otro modo:

<acción(es)>

Fin Segun

C++

```
switch (expresión){  
    case valor1:  
        <acción(es)>;  
        break;  
    case valor2:  
        <acción(es)>;  
        break;  
    default:  
        <acción(es)>;  
}
```

Estructura de Control Iteracion

Se usan cuando una o varias instrucciones deben repetirse muchas veces

Bucle: Conjunto de instrucciones que debe ejecutar

Iteracion: Repetición de un conjunto de instrucciones

La forma de indicar el número de iteraciones define el tipo de bucle (for, while, do-while, repeat-until)

Sentencias de repetición:

Pseudocódigo:

Mientras

Hacer Mientras

Para

 while

Mientras: Se utiliza cuando no sabemos el número de iteraciones que vamos a utilizar

Si da verdad las hace y si no se para

La condición está al inicio del bucle y este se ejecuta siempre que sea verdadera.

Tener mucho cuidado con los bucles que nunca se encuentran o son infinitos.

Pseudocódigo

C++

Nacer Mientras
do-while

Es similar al buckle, pero es post-test

Este tipo de bucle sirve para: **menús**

Pseudocódigo

C++

hacer

<acciones>

Mientras <condicid>

do }

<accion(es)>;

{ while (Condición(es));

Para
for

Se utilizan cuando se sabe el numero de veces que se debe repetir

Utiliza una variable numérica para saber cuantas iteraciones deben realizarse

Pseudocódigo:

Para < inicialización > hasta < condición >
con paso < paso > hacer
< acción(es) >
Fin Para

C++

for (< inicialización >; < condición >; < paso >) {
< acción(es) >
}

Tipos de errores en programación:

- * Compilación
- * Instrucción
- * Lógico (los más difíciles)

Contadores:

- * Variable que se utiliza para contar algo
- * Se usa dentro de un ciclo (normalmente) y cambiamos un valor cambiando la constante.

Acumuladores

- * Variable que se usa para sumar valores
- * Se utiliza normalmente dentro de un ciclo

Actividad pruebas de escritorio

Problema 1

Variables Valores

x	0
a	5
b	5
c	5

Problema 4

Variables Valores

f	2
g	-15
h	-5
x	-20

Problema 2

Variables Valores

c	86
b	25
a	-18
x	15

Problema 5

Variables Valores

c	5
b	25
a	12
x	10

Problema 3

Variables Valores

c	2
x	5
b	3
q	15

Problema 6

Variables Valores

a	7
b	-1
c	3
x	10

Problema 7

Variables Valores

$$\begin{array}{l} x \\ y \end{array} \begin{array}{l} 15 \\ 10 \end{array}$$

Problema 11

Variables Valores

$$\begin{array}{l} z \\ x \\ y \end{array} \begin{array}{l} 35 \\ 10 \\ 4 \end{array}$$

Problema 8

Variables Valores

$$\begin{array}{l} x \\ a \end{array} \begin{array}{l} 34 \\ 6 \end{array}$$

Problema 12

Variables Valores

$$\begin{array}{l} x \\ y \end{array} \begin{array}{l} \infty \\ \infty \end{array}$$

Problema 9

Variables Valores

$$\begin{array}{l} a \\ b \\ x \end{array} \begin{array}{l} 14 \\ 16 \\ \text{No entero} \end{array}$$

Problema 10

Variables Valores

$$\begin{array}{l} x \\ y \end{array} \begin{array}{l} 25 \\ 25 \end{array}$$

Pseint

Azar: genera números de 0 hasta el número que tengas -1

Ej:

Azar(10) : genera números de 0 a 9

Azar(8) : genera números de 0 a 7

~~29x0f~~V ~~2d0d~~V
C++ es C con más cosas

Directivas (solo se incluyen las que necesitas)

exclusiva de c++
 \downarrow

#include <iostream> Para lectura y escritura
using namespace std;

#include <stdio.h> Standard input-output header

#include <conio.h> Console input-output

#include <math.h> funciones matemáticas

Programa principal

1/ directivas → con eso ya se puede compilar
int (main) {

return 0; ← esto es opcional
} ↑ te soltará este número cuando todo salga bien

Siempre que haya un cambio en el código, guardenlo y complílenlo

NO dejen espacios en el nombre del archivo

Diferencias entre "error" y "warning"

error: no lo compila ni genera el archivo ejecutable

warning: si genera el archivo ejecutable y compila, pero te avisa de posibles problemas

Variables

int : enteros char: caracteres

float : decimales, 2 decimales

double : muchas decimales

Para todos los lenguajes de programación:

if : significa el SI de Pseint

else: significa el SINo de Pseint

switch: significa Segun de Pseint

Si pones en C++ las ~~letras~~ del español como la ñ o letras con acentos, te oírás cosas raras, para resolverlo pon:

char('64')

↑

Para poner el arroba, busca en google el código ASCII de la cosa que quieres responder y pon el número

sería así:

Cout << "Introduce un n" << char[163] "mero: "

Análisis: Datos entrada

Datos Salida

Requisitos adicionales/restricciones

Diseño: Lista de tareas

Pseudocódigo

Switch (...) {

Case 1:

($x > break$; ($x \geq 1$))

case 2:

break; ← importantísimo poner
break porque si no
hay bugs, el SI hace que
te saltes hasta el final.

while: es el mientras de Pseint

X++ muestra primero la variable y luego
hace el aumento, sale la pantalla la va-
riable y el aumento lo hace pero no
lo muestra porque lo hace después.

(++X): hace el aumento primero y luego
muestra la variable.

for: es el para de Pseint

ej:

for(x=0; x<5; x++) {

~~do while~~: es el hacer mientras de Print

Se escribe así:

do {

$$(x += y) = (x = x + y)$$

} while ();

Char: guarda sólo 1 carácter

String: guarda palabras, pero lo muesta hasta que haya un espacio por como trabaja el `Cin >>`, para eso usa `getline`.

Para leer muchas palabras que estén separadas por espacio se usa

`getline(Cin, x)` PON LA DIRECTIVA
#Include <string>

La comilla simple es para caracteres (' ')

Las comillas dobles son para cadenas (" ")

el Char puedes poner: char(64)

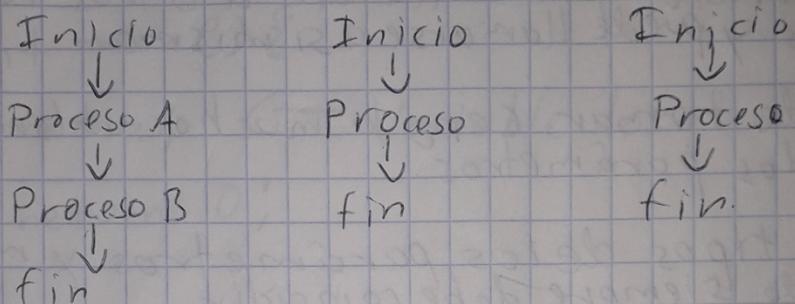
↗

código ascii

Programación Modular

Funciones

Principal Proceso A Proceso B



* Es un bloque de código que realiza una operación

* Puede definir opcionalmente parámetros de entrada que permiten pasar argumentos a la función

* Puede devolver valor **Uno y sólo 1**

* Son útiles para encapsular las operaciones comunes en un sólo bloque reutilizable.

* Deberían tener un nombre que describa lo que hace.

* Acepta datos, realiza procesos, genera resultado

- * Cuando se usa se dice que se llama o invoca
- * Se puede llamar todas las veces que se deseé
- * Una función puede llamar a otras funciones y puede llamarse a si misma.
- * Puede tomar cero o muchos valores llamados parámetros.
- * Los tipos de los parámetros y resultado siempre debe coincidir.
- * Las variables declaradas dentro del cuerpo se denominan variables locales.
- * De forma predeterminada, los argumentos se pasan a la función por valor (copia). También se pueden pasar por referencia.
- * Cuando una función modifica un argumento que se pasa por referencia, modifica el objeto original.

```
#include <iostream>
using namespace std;
int sueldo; ← Variable global
int main() {
    int a; ← Variable local
    return 0;
}
```

Para escribir caracteres reservados, haz esto:

\← esto quiero que me lo muestre

✓ Mostraré este porque la diagonal invertida de atrás hace que la diagonal que está encerrada no de la función y la muestre.

En c++ hay declaraciones y definiciones

Características funciones:

* Las funciones deben tener definiciones

* Incluye

* Si se va a usar una función

Declaración: Se indica el tipo de valor que devuelve, al final se pone ; (vá antes del int main)

int suma (int a, int b);
 ↑ ↑
 Puede o no tener esto

Definición: escribir el código que va a tener esta función (va después del int main)

int suma (int a, int b){
 return a+b;
 ↑ ↑
 Debe si o si tener esto

Usa llaves de key.

```
if(x>0)  
cout << "vale 0" <<
```

esto funciona aunque no tenga llaves
porque sin llaves solo jala la linea
de abajo, no mas lineas.

`System("Pause")`: hace que tengas
que presionar una tecla para que con-
tinue (cualquier tecla)

`System("cls")` limpia lo que hay
en pantalla cuando se está ejecutando.

¿Cómo hago para declarar una buena
función?

Con estas preguntas!

¿Qué tipo de dato deberá de volver?
¿Cuál será un buen nombre?
¿Qué datos debe recibir?

¿Cómo generar números aleatorios
en C++?

función "rand"

debes incluir en el código la directiva
`#include <cstdlib>`

/

se escribe así:

```
int rand(void);
```

e) empleos

VI = rand () % 100; // genera numeros de 0 a 99

$$\checkmark 1 = \text{vanda } (\%) / 100, \text{ il genera } \dots \dots \dots$$

$$\checkmark 2 = \text{vanda } (\%) / 100 + 1; \text{ il genera de la } 100$$

$\text{V2} = \text{rand}() * 100 + 1$, genera de 1 a 100
 $\text{V3} = \text{rand}() \% 30 + 1985$, genera de 1985 a 2014

strand (time (NULL)); // esto cambia la semilla para que cuando ejecutes el programa no salgan siempre los mismos numeros aleatorios.

Ponlo así en el código

int Numeral;

`sound(time(NULL));` ← Para que no salgan los
mismos mensajes de error.

Numeros = rand()%100 Para generar numeros de 0 a 100

función void

void(x){

} return; ← Como es "void" (vacío) no
regresa nada en return

La función main se tiene que escribir:
Tiene que ser a fuerzas ser int.
int ← main ()

main()

#include <stdlib.h>

esta librería hace que puedas usar:

EXIT_FAILURE; -- da 1
EXIT_SUCCESS; -- da 0

Parámetros de int main:

int argc; cantidad de argumentos

→ Se puede poner cualquier valor y valen lo mismo, pero es común ponerlos con esos nombres argc...

char* argv[]; donde se guarda

No se usan mucho esos parámetros

Solo se pueden poner esos 2 parámetros, ~~pues~~ en orden, y se tienen que poner los 2 siquieres usandos.

NO HAY MAS PARÁMETROS EN C Y C++ PARA MAIN

Char Nombre[20]

Puedes escribir la cantidad de letras que hay entre los corchetes y los muestra

Las funciones se pueden llamar igual:

int Suma(int)
int suma(int,int)

Son lo mismo pero fíjate que tienen distinta cantidad de variables.

Vario.

Constante: Siempre va a tener el mismo valor, nunca va a cambiar.

así se pone: const int = 0

Tienes que poner el valor a la fuerza

Es recomendable ponerlos en mayúsculas
+ todos

const int PI

const int EULER

const int DIAS_SEMANAS

strcmp(x, y)

hace esto:

<0 si el primer dato es menor

0 si son iguales

>0 si el primero es diferente

alicia

Alan

Daria 1

alicia

alan

daria 1

Alan

Alan

Daria 0

Alan

alan

Daria -1

Rewerda el ACSII

Cuando el compilador busca variables, busca primero las locales y luego las globales.

Pase de parámetros por valor.

Hacer copia del valor de una variable a otra

Cuando lo modificas sólo afecta a uno

Por referencia

Consiste en proporcionar la función a la que se le quiere pasar el argumento la dirección de memoria del dato.

Cualquier modificación afecta a los que se usan, porque están en la misma parte de la memoria, no se usa copia como en el pase de parámetros por valor.

cout << endl; // sale 0xXXXXX

↑

¿Quieres ver en qué posición de la memoria RAM está la variable? ¿O cosa que hay puesto?

Pon – antes de la variable

Parámetros por referencia:

int(int, & int)
↑

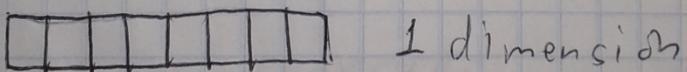
Anderson para que sea por referencia.

Cadenas, Vectores y Matrices

Estructuras de datos

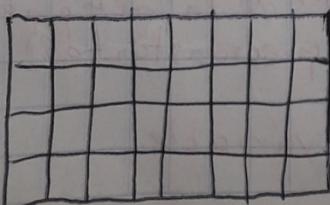
colección de datos que se pueden caracterizar por su organización y las operaciones que se definen en ellas.

Vector:



1 dimensión

Matriz:



2 dimensiones

La computadora puede representar n dimensiones, pero para los seres humanos es muy difícil representar más de 3.

Todos los datos empiezan en 0.

Vector

1	2	3	4	5	6	7	8
0	1	2	3	4	5	6	7

↓
índice

elemento.

Operaciones comunes con vectores

Asignación

Lectura y escritura

Recorridos

Búsquedas

Ordenamiento.

Asignación:

int edades [5];

Cuántas edades vas a guardar
No puedes poner números negativos.
porque si lo haces afecta la computadora.

edades [0] = 15;

posición en la que vas a guardar
el dato (empezarán desde 0)

↑ dato que vas a guardar

puedes poner variables tambien aparte de constantes

Lectura y escritura

La escritura implica pedir

¿Cómo dar valor inicial a un arreglo?

int edades [5] = {1, 2, 3, 4, 5}

↑↑↑↑↑
 Valor
1 2 3 4 5
 ↑
 posición

Eso se usa para cuando tenemos que poner datos a todas las posiciones del arreglo

int edades [5] = {1, 2}

↑

posición 1 = 1
 1 = 2
 1 = 0
 1 = 0
 1 = 0

Pon.

int edades [];

da error

int edades [] = {2,5};

esto sería como [2] = {2,5} ya que aunque no pusiste número dentro de los corchetes, como pusiste 2 datos entre las llaves, como son 2 entonces la compiladora convierte [] a [2].

¿Cómo mandar arreglos por función?

- Void imprimir arreglo {int [], int};

Con esto sabe que recibirá arreglo

Para ponerlo en main

int main () {

imprimir arreglo (números, 5);

}

no se pone entre corchetes

Depuración: Prueba de escritorio en la computadora.

Ordenamiento:

Ordena datos
métodos!

C# no es la evolución de C++

C# es competencia de java

C# es de microsoft.

- * Bubble sort
- * Insertion sort
- * Selection sort
- * Shell sort
- * Merge sort
- * Quick sort.

Cada espacio del array
es 1 byte, 1 byte, 2 bytes
int x [1] int x [2]

ARREGLOS DE CARACTERES EN C ES LA MUERTE.

¿Cómo poner arreglos con caracteres?

char nombre[10] = {'A', 'l', 'o', 'n', 'g', 'o', ' ', 'O'};

Puedes poner '\0' o el cero sin comillas.

Fin de cadena \0

Para poner todo en orden ponemos \t
ej.

Sin \t

1x2 1x3 1x4 1x5
1x3 1x2 1x4 1x5
1x4 1x5 1x3 1x2

Con \t

1x2 1x3 1x4 1x5
1x3 1x2 1x4 1x5
1x4 1x5 1x3 1x2

char Saludo1 [] = { 'H', 'O', 'L', 'A', '\0' };

char Saludo2 [5];
Saludo2 [0] = 'H';
Saludo2 [1] = 'O';
Saludo2 [2] = 'L';
Saludo2 [3] = 'A';
Saludo2 [4] = '\0';

char Saludo [5] = "Hola";

Así se pone Hola

Operaciones comunes con arrays con caracteres

Calcular longitud

Comparaciones

Concatenaciones

Extracción de subcadenas

Búsqueda de información

Calcular longitud

#include <string.h>

tamaño = strlen (nombre);

variable función que dice cuantas letras
para agarrar tiene el arreglo
el valor de la función

Comparación

"Emilio" == "Emilio" True
"Emilio" == "Emilia" False

"Garcia" < "Gomez" $6 = 6, A < O$
Verdadero

strcmp (cadena1, cadena2)

Concatenacion

Reunir varias cadenas en una

#include <string.h>

```
char cad1[] = "Hola";
char cad2[] = "mundo";
char cad3[30];
```

```
strcpy(cad3, cad1);
strcat(cad3, cad2);
strcat(cad3, "desde C++");
```

```
cout << cad3 << endl;
```

```
strcpy(cad3, cad1);
```

Copia cad1 a cad3 pero borra lo que
tenia cad3

```
strcat(cad3, cad2);
```

Copia cad2 a cad3 pero sin borrar
lo que tenia cad3.

El string no funciona en C

Cadenas Es la forma fácil de hacer arreglos de caracteres

Son objetos que representan secuencias de caracteres

Se utiliza la clase string para crear las.

Declaración: String x;

String x = "Hola";

getline(cin, x); para leer todo y no separa cuando hay un espacio.

Para arreglos de caracteres

cin.getline(nombre, 50); lee todo lo que le pongas y no se para cuando hay un espacio.

¿Cómo concatenar string?

x = "X" + "Y" + "Soy" + "X";

x = x + y;

Métodos: cosas que se pueden aplicar a clases para que hagan cosas específicas.

str.size() // esto dice el tamaño de
 ↑ ↑
 variable método

Los métodos solo se pueden poner en objetos

El objeto es lo más abstracto dentro de la programación

¿Cómo convertir strings a número?

int entero

String x = "2001";

int numero = stoi(x)

¿Cómo convertir int, float, double a string?

to_string(x);

while (true) {

} Esto siempre se va a ejecutar

while (false) {

} Esto nunca se va a ejecutar.

Palabra == (palabra Volteada)? return true;
return false;

Otra forma de poner esto:

if (palabra == palabra Volteada) {

 return true;

}

else {

 return false;

}

Matrices

- Vector de vectores
- Todos los elementos son del mismo tipo
- Necesita dos indices para identificar cada elemento del array
- El primer indice se refiere a la fila y el segundo a la columna
- Tipo dato matriz [fila][columna];

• ¿Cómo inicializar una matriz?

int matriz[3][3]
 ↑ ↑
 filas columnas

int matriz[3][3] = { {1,2,3}, // fila 1
 {4,5,6}, // fila 2
 {7,8,9} }; // fila 3

Función trim: elimina los espacios
al inicio y al final de una palabra

trim(" chistian ") da:
"chistian"

cin.ignore()

esto ignora lo que hay pendiente en el cin.

cin>> "tri no mio";
cin.ignore

Si hay esto:

cin>> X;
getline(cin, Y);

puede haber bugs, así que ponlo así

cin>> X;
cin.ignore();
getline(cin, Y);

String

String.substring(0, x)

↑
Agarra los valores que están de la posición
0 a X.

PROGRAMACIÓN ORIENTADA A OBJETOS

Un paradigma es una forma específica de resolver problemas computacionales

La programación orientada a objetos permite modelar los problemas de la vida cotidiana de manera sencilla.

Se basa en el uso de objetos

Técnicas: herencias, abstracción, polimorfismo, encapsulamiento

Un objeto es la parte modular, y es todo ello que podemos percibir por nuestros sentidos

Primero hay que ver si hay un objeto e identificarlo

¿Por qué utilizar POO?

El string es un ejemplo.

```
int edad;
string nombre;
char sexo;
string dirección;
```

{ para 1 persona

```
edad1, edad2...
nombre1, nombre2...
char1, char2...
dirección1, dirección2...
```

{ para 2 personas
o más

¿Por qué no crear una variable tipo persona?

¿Es mejor no?

La POO permite crear tus propias variables, tus propias clases, tus propios datos

• ¿Cómo escribirlo?

```
class Persona {  
    int edad;  
    string nombre;  
    char sexo;  
    string direccion;  
}
```

```
int main () {
```

```
    Persona christian, pedro, persona3;
```

```
}
```

• ¿Cómo utilizar más de 1 archivo?

en donde está ma-

en la clase

Persona.cpp

```
#include "Persona.cpp"
```

Otra cosa

Si en la directiva pones
" < >" busca el archivo
en la carpeta de ct

si le pones

Lenguaje Orientado a objetos: Lenguaje con el que podemos trabajar con los conceptos de POO

C++ es orientada a objetos y estructurada

JAVA es orientada a objetos y funcional

La orientación a objetos no solo aplica al lenguaje de programación, se viene aplicando también en el análisis y el diseño.

Modelo orientado a objetos

- * Objetos
- * Clases
- * Herencia
- * Envío de mensajes
- * Abstracción
- * Encapsulamiento
- * Ocultación
- * Polimorfismo

Objeto

Cualquier cosa que pueda ser representada en el mundo real, como silla, coche, etc.

La clase objeto está omisa de todo

Si tienes un servidor con interfaces gráficas o le conectas una pantalla, consume recursos, por eso los servidores no suelen estar conectados a monitores y no tienen interfaces gráficas

Todos los objetos tienen 2 cosas:
Características y comportamiento.

Estado / los atributos

Un objeto es una unidad de código compuesto de variables y métodos relacionales.

Diagrama UML

Automovil

-marca: string } características o
-modelo: string } atributos
-color: string }
-velMax: int }
+ frenar(): void } comportamiento
+ acelerar(): void }
+ retroceder(): void } métodos

el símbolo de - en UML en un atributo: es privado

y el + significa público

en la clase cuáles son los atributos y características

el objeto ya tiene valores

Clase

Automobil

Marco: String

Modelo: string

Color: string

$\nabla \phi_h : \mathbb{R}^d$

↑
flmolde

Objeto

Marca = ford

Modelo: xix

Color: René

vph: 500 Km/h

La clase convalecidos

```
class Persona {  
    int altura;  
    string nombre;  
    string edad;
```

La clase

Personaje René, Chomaí
 ↓
Clase Objetos

en UML

-) privado
- + : público
- # : protegido

(crea una instancia de la clase persona
esto mismo a decir:

Crea un objeto de la clase persona

instancia es sinónimo de objeto

¿Cómo definir una clase en c++?

class NombreClase {

class gato {

Atributos (características)

Métodos (comportamiento)

31

30

Atributos: tipo entero, string, etc

Métodos', como una función

```
class Persona {
```

Private:

```
    int edad;
    string nombre;
    char sexo;
    string dirección;
```

Atributos

Public:

```
void saludar() {
    cout << "Hola a to2" << endl;
}
```

Método

}

Todos los atributos y métodos en C++ por defecto son privados

Acceder a elementos

```
int main() {
```

```
    Persona P1;
```

```
    P1.saludar
```

this -> para imprimir los datos de la clase

Heredencia

Uno de los conceptos más cruciales de la POO

Consiste en que una clase puede heredar
atributos y métodos a varias subclases

Esto significa que una subclase, aparte de los atributos y métodos propios, tiene incorporados los atributos y métodos heredados

Celular

- marca: string
- modelo: string
- + llamar (): void
- + colgar (): void

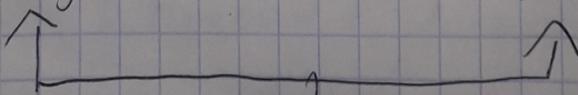
flecha vacía apuntando al papá

Celular_nuevo

- precio_venta: int
- fecha_ingreso:

Celular_malogrado

- motivo: string
- dueño: String



incluidos todas las cosas de Celular

Abstracción: captar características esenciales de un objeto, así como su comportamiento

Ocultamiento: Oculta detalles del comportamiento de una clase, permite 2 cosas: restringir y controlar el uso de la clase

Todos los elementos, o sea variables y los métodos que sean privados, sólo se pueden manejar desde la clase

Los protegidos sólo pueden ser accedidos de la clase y desde las clases hijas.

Herencia:

```
class Animal { } } clase padre
```

```
class Perro: public Animal { } } clase padre      } clases hijos  
class Gato: public Animal { } }
```

Polimorfismo: 2 cosas se pueden llamar igual y hacer cosas distintas pero en clases distintas

Arreglo [].x

Se puede hacer esto: $\text{José} \cdot x[1].\text{pedir}()$
más de 1 punto.

fas7

Constructor: método de una clase que se ejecuta cuando se crea un objeto.
No tiene tipo de dato

Se llamará de la misma forma que la clase

Class Empleado
Public
Empleado () { } } constructor.

Empleado () { }
SueldoPorHora = 10; } } cualquier objeto que utilice
nombre = "Indefinido"; } } estos métodos, tendrá estos
valores } }

No pongas cin y cout en constructores

Empleado() {
 } } Constructor - sin parámetros

Empleado (Nombre) {
 Sueldo = Nombre;
}; Constructor con parámetros
3

Puedes construir más de 1 constructor sólo si tienen diferentes parámetros

Empleado ()
Suelo = 13°;

```
    inf main() {  
        Empleado yoyo, yejo;
```

$$\text{empleado} \quad (\times) \quad \{ \\ \text{sueldo} = x; \\ \}$$

Van a valer 1)
los 2

Vana valer 3y 8-7

```
int main () {
```

3

Si creas uno o más constructores, que no se te olvide crear el constructor por defecto.

Ej C) {

§

el por defecto es el que no tiene parámetros

Getter: Para cuando un atributo es privado, no se puede acceder desde afuera de la clase.

Es un método, no recibe nada, devuelve la cosa del mismo nombre.

private:

int float;

public: tienen que escribirle get

int getfloat() {
 return float;
}

• Modificar atributos (sets)

Todos los métodos sets son void y si reciben parámetros

```
class gato {  
    private:  
        int chueldo;  
  
    public:  
        void setChueldo(int a) {  
            chueldo = a;  
        }  
}
```

private:

→ float sueldo por hora;

public:

```
void setSueldoPorHora(float sueldoPorHora) {  
    this->sueldoPorHora = sueldoPorHora;  
}
```

↑
3

agregale this para que esto aga me esto,

Métodos fuera de la clase

```
class gato {  
    private:  
        string color;  
        int vidas;  
    public:  
        gato (String); } Declaraciones como en las fun-  
        void miau(); } ciones  
}; Agregue el nombre de la clase y 2 veces {}  
gato::gato (String c) { } El método fuera de la  
}; clase.
```

Prácticamente es como las funciones

Uml

Uml dice cómo crear documentaciones o diagramas y esquemas

No es un lenguaje de programación

En Uml tiene diagramas:

Casos de uso

clases

secuencia

colaboración

estados

Actividad

Paquetes

Arquitectura de software

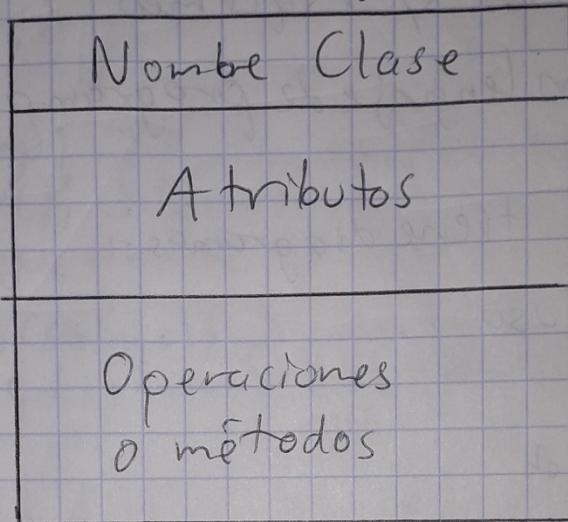
etc.

Diagrama de clases

Estructura estática que hace entender rápidamente la relación de todas las clases, incluyendo herencia y más.

Elementos diagrama de clase

En uml es representada por un rectángulo dividido en 3 partes



Atributos! variables

+ = públicos

- = privados

= protegidos

Métodos! acciones

Las cosas de la documentación se le llaman artefactos, como los diagramas de clase, casos de uso, etc.

Se usa artefactos porque decir "documentación" da hueva

Simbología

Asociación: relación de una clase y otra. NO ES UNA RELACIÓN FUERTE (o sea es una relación débil)

Relación fuerte: si se rompe algo, lo que esté relacionado con esto se rompe también

Relación débil: si se rompe algo; lo que esté relacionado no afecta, queda intacto.

Cardinalidad: indica el grado y nivel de dependencia de clases

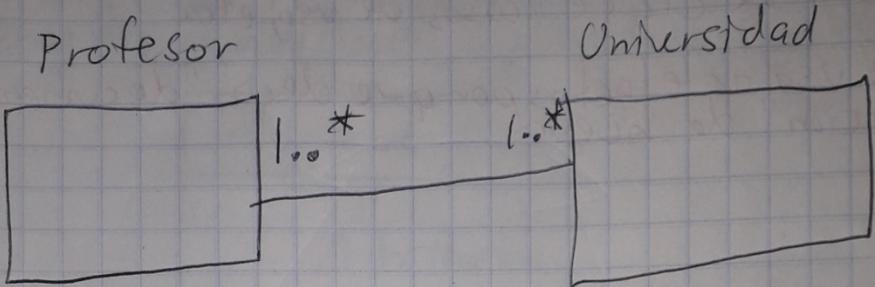
* = Cero, uno o más
0,1 = Cero o uno

1^{*} = Uno o más

1 = Solo 1

1..5 = está entre 1 y 5

Ej cardinalidad:



Un profesor puede trabajar en mínimo 1, máximo las que pueda.

Una universidad puede tener mínimo 1, máximo los que pueda.

Agregación/Composición:

Una clase está construida por otras clases más.

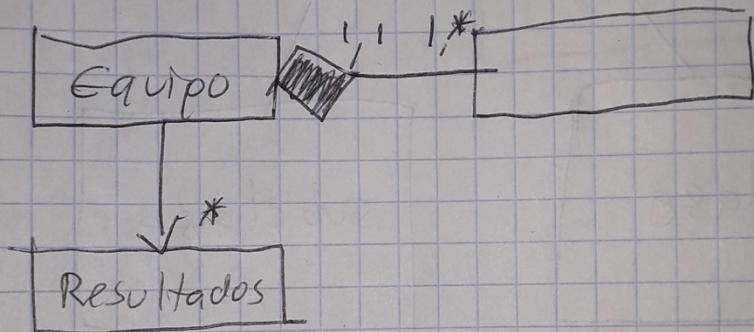
Por valor 

Esto es composición, si se rompe algo, lo que está relacionado con el se rompe también

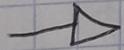
Por referencia  →

Si algo estare relacionado con el, si se rompe,
a lo que este relacionado no afecte.

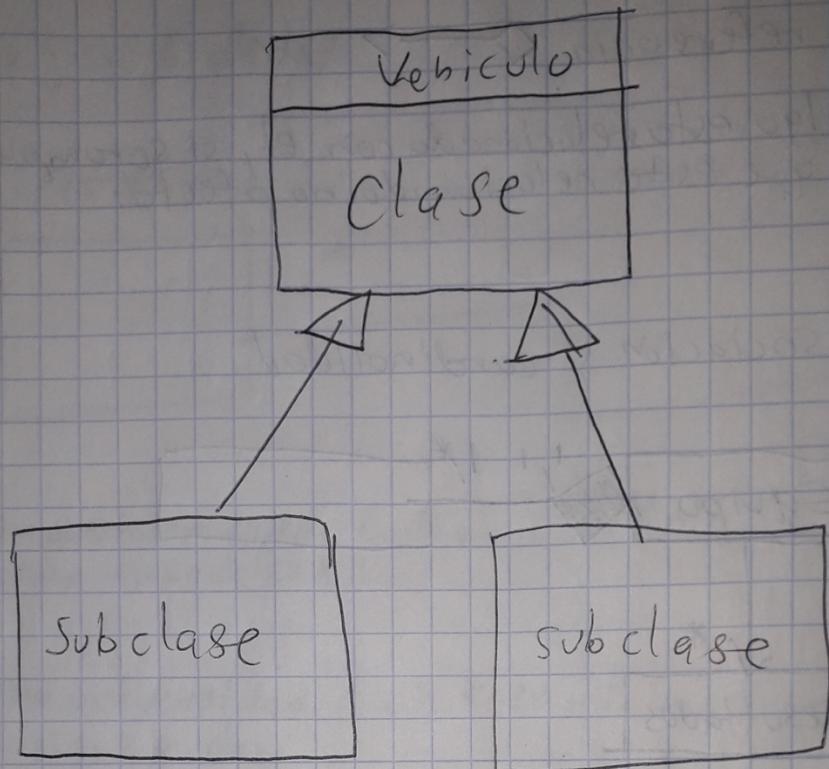
Asociacion y cardinalidad



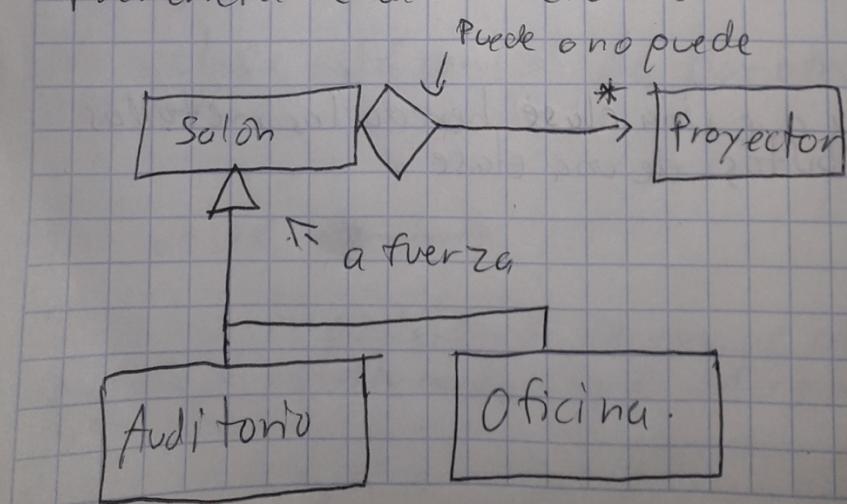
Herencia (especialización/generalización)



Indica que una clase hereda los métodos
y atributos de una clase

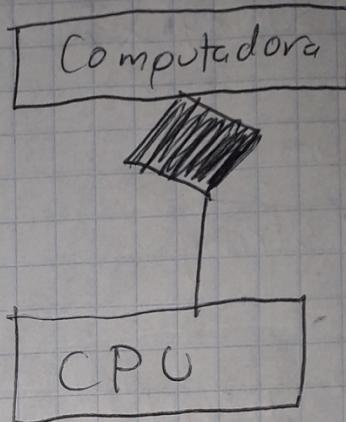
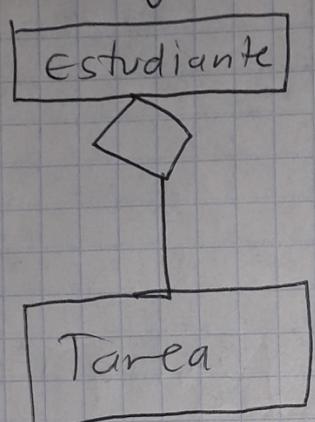


Herramienta de asociaciones



Norma

Agregación vs composición



Ventajas de UML

- * Genera código automáticamente
- * Es el más usado
- * Solución a errores
- * Relaciones clases y sistema
- * Se protegen datos
- * Componentes de sistema
- * Se posibilita una reducción de acoplamiento
- * fuente de generación de código
- * forma más fácil de ver todas las clases.

Cuando pones variables en c++

string x;

en uml:

x: string

Cuando creo código con el uml, quítale los voids a los constructores

En strings siquieres poner muchas cosas:

"x" + x + "y" + y + "z" + z;
}

Se usan los +

Cuando hay esto:

if(x)

Suele ser un boli, por que evalua si es true