UTN FRGP
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL GENERAL PACHECO

PROGRAMACIÓN I

TUP - 2025 - 1C - Turno noche - Presencial
Comisión 111

¿CUÁNDO Y DÓNDE?

Martes

18:00 a 22:00

Diagramación

Aula 201

Viernes

18:00 a 22:00

Codificación

Laboratorio 4

DOCENTES

Diagramación

- **Maximiliano Sar Fernández** - msarfernandez@docentes.frgp.utn.edu.ar
- **José Alberto Rodríguez** - jarodriguez@docentes.frgp.utn.edu.ar
- **Alejandro Martín Gómez Nieto** - alejandro.gomez@alumnos.frgp.utn.edu.ar

DOCENTES

Codificación

- **José Alberto Rodríguez** - jarodriguez@docentes.frgp.utn.edu.ar
- **Alejandro Martín Gómez Nieto** - alejandro.gomez@alumnos.frgp.utn.edu.ar
- **Javier Agustin Larroca** - javier.larroca@alumnos.frgp.utn.edu.ar

CAMPUS VIRTUAL

- Novedades
- Cafetería
- Foro de dudas generales
- Guía de cursada
- Equipo docente
- Repositorio
- Material de cada unidad
- Ejercicios de cada unidad

https://frgp.cvg.utn.edu.ar/

- **Usuario:** DNI.frgp
- **Clave:** DNI

CRITERIOS DE APROBACIÓN

Diagramación

- **Parcial 1** - Se aprueba con 6 o más, cuenta con su instancia de recuperación.
- **Parcial 2** - Se aprueba con 6 o más, cuenta con su instancia de recuperación.

Codificación

- **Tareas entregables** - Al finalizar cada guía de ejercicios se dará una tarea grupal a realizar en clase y entregar.
- **Trabajo Práctico Integrador** - Aplicación que deberán desarrollar de manera grupal y posteriormente defender. Tendrá un tiempo asignado para su desarrollo, resolver dudas y trabajar en el avance y una defensa. El TPI tiene una instancia de recuperación si su defensa no fuera satisfactoria.

CONTENIDOS

Etapa 1

1. Estructuras secuenciales

- Variables y constantes
- Operadores

2. Condicionales

- Simple
- Múltiple

3. Ciclos

- Ciclo exacto
- Ciclo inexacto

4. Ciclos combinados

Etapa 2

5. Funciones

- Diferentes tipos de parámetros

6. Vectores

- Carga y proceso
- Ordenamiento

7. Matrices y Struct

Proyecto final

HERRAMIENTAS PARA LA RESOLUCIÓN DE PROBLEMAS

Diagramación

Mediante diagramas de flujo, deberán ser capaces de representar la solución lógica a situaciones problemáticas.

Codificación

Mediante el lenguaje de programación C++, deberán ser capaces de desarrollar programas que ofrezcan soluciones a situaciones problemáticas.

SOFTWARES

Entorno y compiladores

CODEBLOCKS 20.03
WINDOWS Y LINUX

ZINJAI
WINDOWS

VISUAL STUDIO COMMUNITY
WINDOWS

DEV C++
WINDOWS

VISUAL STUDIO CODE + PLUGIN
WINDOWS, LINUX Y MAC

XCODE
MAC

¿CON CUÁLES TRABAJAREMOS?

C++

Lenguaje de programación

Codeblocks

Entorno de trabajo y compilador

Github

Repositorio y controlador de versiones

¿QUÉ ES PROGRAMAR?

¿Es lo mismo programar que codificar?

PROCESO DE DESARROLLO DE UN PROGRAMA

01 Análisis del problema

Se analizan las características del problema. Se determinan los datos clave de entrada y salida.

02 Diseño de solución

Se determina en detalle las estructuras de programación que se utilizarán. Se hace un esquema que resuelva claramente el problema.

03 Codificación

Se procede a codificar la solución.

PROCESO DE RESOLUCIÓN DEL PROBLEMA

Determinar cuántos y cuáles son los datos de entrada de nuestro programa.
Ponerles un nombre y determinar su tipo de datos.

El algoritmo ideado debe poder transformar los datos de entrada en la información de salida.
Se apreciará que el algoritmo sea eficiente en su resolución.
Resolviendo el problema de la mejor manera y con el menor costo de recursos posibles.

La información de salida debe ser clara, prolija e informar estrictamente lo necesario.

DISEÑO DE UN ALGORITMO

Estructura de datos

Variables simples o complejas, vectores, matrices, etc.

+

Estructura de programación

Decisiones, estructuras de repetición, funciones, etc.

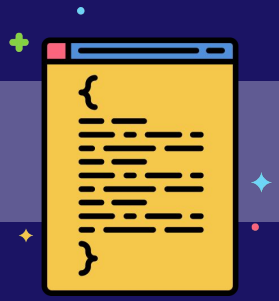
→

Esquema del algoritmo

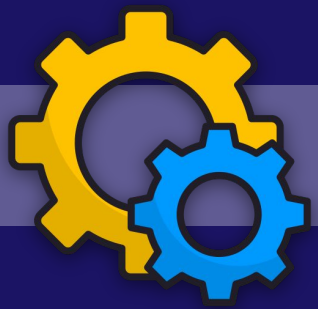
Secuencia de instrucciones precisas y finitas que resuelven un problema en particular. El conjunto de instrucciones en un lenguaje determinado se llama código fuente.

DEL CÓDIGO AL PROGRAMA

- El proceso de transformar un **código fuente** en un **programa** que contiene las instrucciones que sean comprensibles por la computadora se llama **compilación.**



Código fuente



Compilación

Programa

01

SECUENCIALES

CODIFICACIÓN

Cada elemento que utilicemos en un lenguaje de programación debe estar sujeto a una estricta sintaxis. Los elementos que el lenguaje admite son:

- **Variables y constantes**
- **Operadores**
- **Expresiones**
- **Palabras reservadas del lenguaje**

Como muchos lenguajes, C y C++ son case-sensitive. Esto significa que hace diferencias entre mayúsculas y minúsculas.

PALABRAS RESERVADAS DEL LENGUAJE

Palabras que el lenguaje utiliza para identificar tipos de datos, estructuras de programación, etc. Tienen un significado especial para el lenguaje y no pueden ser utilizados como identificadores de nombre.

Propósito	**Palabras reservadas**
Tipos de datos	bool, int, float, char, short, void, long, double
Elementos de programación	if, else, switch, default, break, for, while, do, return, auto, struct, class, static, virtual
Operadores	new, delete, sizeof

VARIABLES Y CONSTANTES

Representación simbólica de espacios de memoria. Es donde se almacenan los datos en procesamiento.

Una variable se identifica con un **tipo de dato** y un **identificador de nombre** (los elige el programador), y permite escribir un dato en la memoria o leer un dato de la memoria. Se puede modificar su valor las veces que sea necesario.

Una constante se identifica con la palabra reservada ***const***, un tipo, un nombre (lo elige el programador), y un valor que no puede ser modificado durante el transcurso del programa en ejecución.

VARIABLES Y CONSTANTES

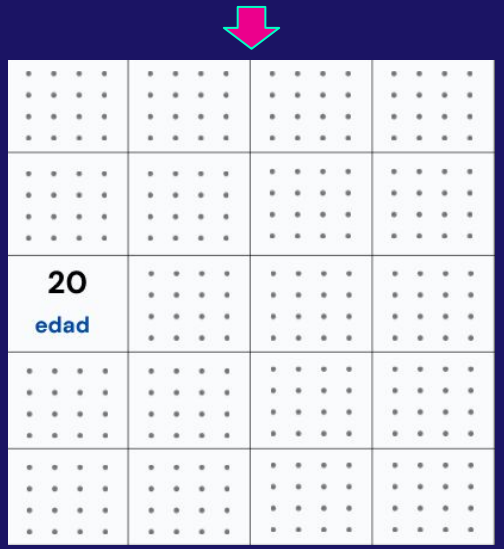
Ejemplos de declaraciones de variables:

- `int legajo;`
- `float sueldo;`
- `string nombre = "Programación I";`
- `string apellido;`
- `char letra;`

Ejemplos de declaraciones de constantes:

- `const int EDAD_MIN = 18;`
- `const float IMPUESTO = 10.5;`
- `const char PAIS = 'A';`

```
int edad;
edad = 20;
```



EXPRESIONES

Conjunto de variables, números y operadores ordenados de acuerdo a las reglas sintácticas establecidas en el lenguaje de programación. No realizan acciones por sí mismas, solo calculan un resultado.
Tiene como objetivo la construcción de instrucciones para la resolución del problema (o parte del problema) planteado.

Ejemplos:

```
50 + 100
aux - 20
10 > 3
"Hola, " + nombre
```

💡 **INSTRUCCIÓN**

Unidad completa de código que realiza una acción. Puede contener una o más expresiones.

```
int a = 5;
B = 50 + 100;
if(10 > 3)
    cout << “Es menor”;
```

OPERADORES MATEMÁTICOS

Necesarios para realizar cálculos matemáticos. Los paréntesis tienen el mismo efecto que en la matemática en las expresiones algebraicas. Sin embargo, en programación no se utilizan corchetes ni llaves para la separación de términos.

Operador	Operación
+	Suma
-	Resta
*	Producto
/	División real
%	Resto de la división entera

Actividad

((2+3)*5)+10
2+3*5+10
5/2
5.0/2
5%2

OPERADORES MATEMÁTICOS

Necesarios para realizar cálculos matemáticos. Los paréntesis tienen el mismo efecto que en la matemática en las expresiones algebraicas. Sin embargo, en programación no se utilizan corchetes ni llaves para la separación de términos.

Operador	Operación
+	Suma
-	Resta
*	Producto
/	División real
%	Resto de la división entera

Actividad

((2+3)*5)+10 → 35
2+3*5+10 → 27
5/2 → 2
5.0/2 → 2.5
5%2 → 1

OPERADORES DE ASIGNACIÓN

Nos permite asignar la expresión que se encuentra a la derecha del operador en la variable que se encuentra a la izquierda.

Formas correctas:

- `edad = 50;`
- `a = b;`
- `nombre = “Jose”;`
- `caracter = ‘Y’;`
- `precio = cant * pu;`
- `cont = 0;`
- `cont = cont + 1;`
- `cont = 0 + 1`
- `cont++;`

Errores de sintaxis

- `50 = 50;`
- `50 = edad;`

Semánticamente incorrecto.

- `edad = edad;`

EJERCICIO

Hacer un programa para ingresar por teclado la cantidad de horas trabajadas por un operario y el valor que se le paga por hora trabajada y listar por pantalla el sueldo que le corresponda.

1 Datos de entrada

2 Proceso

3 Datos de salida

EJERCICIO

Hacer un programa para ingresar por teclado la cantidad de horas trabajadas por un operario y el valor que se le paga por hora trabajada y listar por pantalla el sueldo que le corresponda.

1 Datos de entrada

-Horas trabajadas
-Valor de la hora

2 Proceso

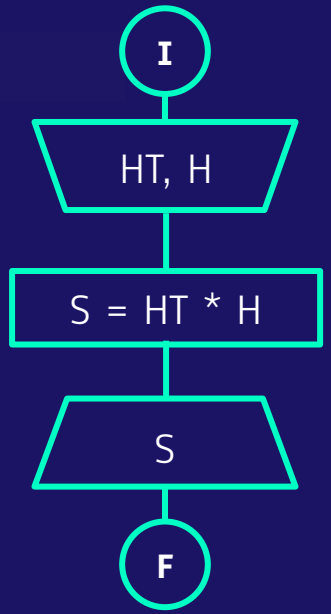
-Calcular el sueldo

3 Datos de salida

-Sueldo correspondiente

DIAGRAMACIÓN

Hacer un programa para ingresar por teclado la cantidad de horas trabajadas por un operario y el valor que se le paga por hora trabajada y listar por pantalla el sueldo que le corresponda.



DIAGRAMACIÓN

01	DECLARAR LAS VARIABLES NO ES OBLIGATORIO	`float valorHora, horas, sueldo`
02	INGRESAR EL PRIMER NÚMERO POR TECLADO	`valorHora`
03	INGRESAR EL SEGUNDO NÚMERO POR TECLADO	`horas`
04	REALIZAR PRODUCTO	`sueldo = valorHoras * horas`
05	MOSTRAR POR PANTALLA EL RESULTADO	`sueldo`

CODIFICACIÓN

Hacer un programa para ingresar por teclado la cantidad de horas trabajadas por un operario y el valor que se le paga por hora trabajada y listar por pantalla el sueldo que le corresponda.

```
int main()
{
    int horastrabajadas;
    float valorHora, sueldo;

    cout << "Ingrese la cantidad de horas trabajadas: ";
    cin >> horastrabajadas;
    cout << "Ingrese el valor por hora trabajada: ";
    cin >> valorHora;

    sueldo = horastrabajadas * valorHora;

    cout << "El total a cobrar es: $";
    cout << sueldo;

    return 0;
}
```

CODIFICACIÓN

01	DECLARAR LAS VARIABLES OBLIGATORIO	`float valorHora, horas, sueldo;`
02	INGRESAR EL PRIMER NÚMERO POR TECLADO	`cin >> valorHora;`
03	INGRESAR EL SEGUNDO NÚMERO POR TECLADO	`cin >> horas;`
04	REALIZAR PRODUCTO	`sueldo = valorHora * horas;`
05	MOSTRAR POR PANTALLA EL RESULTADO	`cout << sueldo;`