Centro de
Investigación en
Computación

Sesión 1

Lenguajes de Programación

Sintaxis y Semántica

Estructura y Significado

de un Programa

Course

Curso de Selección

Instructor

Acosta Bermejo Raúl

Lecture notes

2025-A

Marzo-Abril del 2025

Instituto
Politécnico
Nacional

Table of contents (outline)

Tabla de contenido

1. Conceptos y técnicas de lenguajes de programación
   i. Lenguajes más representativos
   ii. Estructura léxica, sintáctica y semántica

Introduction

Introducción

Paradigmas de programación

- Imperativos: Lenguaje C, Pascal, Fortran, Basic.
- Funcionales: Lisp, Scheme (Bigloo dialecto)
- Lógicos: Prolog
- Híbridos con nuevos conceptos
  - Programación Orientada a Objetos: Imperativo + OO

    Ejemplos

    C + OO = C++ Imperativo

    Caml + OO = Ocaml Funcional
  - Paralelismo/Concurrencia

    Parlog = Prolog + Paralelismo.

    Haskel = Funcional + Paralelismo

Introduction

Introducción

Lenguajes especializados

- Scripts
  - Bash: archivos, comandos.
- Bases de datos
  - dBase[1971], SQL[1974]., Primer estándar SQL[86].
  - Store procedures, triggers, etc.
- Para aplicaciones web
  - HTML, CSS, JavaScript
  - PHP
- Matemáticas
  - Matlab, Mathematica, Maple, Octave, Scilab, etc.
- Graficos o Simulaciones
  - Simulink[Matlab], utilizan bloques (grafos).

Para representar datos

CVS

XML

Json

Procesar texto

Perl

Lenguaje C

Conceptos básicos

Sintaxis

Semántica

Lenguaje C

Intro

El desarrollo inicial de C se llevó a cabo en los Laboratorios Bell de AT&T entre 1969 y 1973; según Dennis Ritchie, el periodo más creativo tuvo lugar en 1972.[1] Se le dio el nombre «C» porque muchas de sus características fueron tomadas de un lenguaje anterior llamado «B». Wikipedia.

Estructura

Como se construye un programa, una aplicación?

Los fundamentales:

1. Archivo (s) con extensión
2. Palabras reservadas
3. Constantes y Variables: identificadores
4. Instrucciones
5. Librerías
6. Pre-procesador: Macros
7. Comentarios

```c
/* Suma de n números */

#include <stdio.h>
int main( ) {
    int num=0,suma=0;

    do {
        suma=suma+num;
        printf("un número: ");
        scanf("%d",&num);
    } while(num>=0);
    printf("suma es: %d",suma);
    return 0;
}
```

Lenguaje C

Intro

Sintaxis

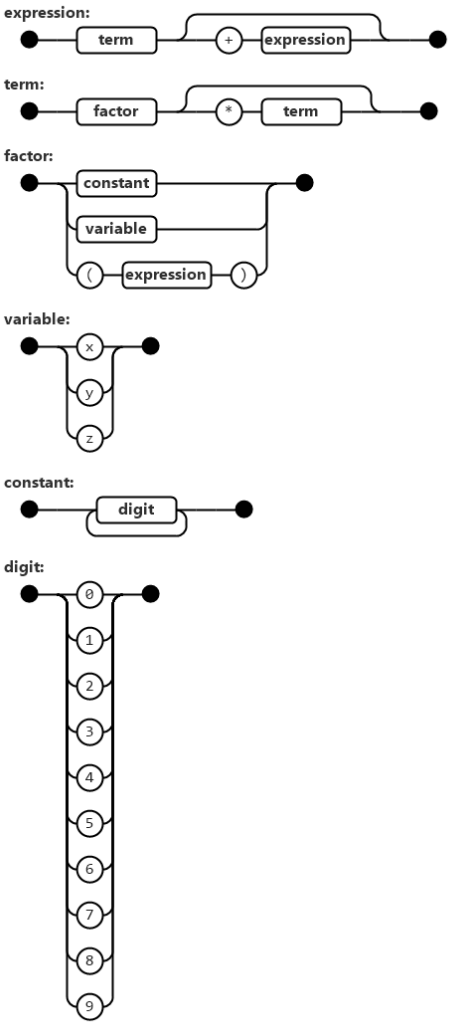
```
<expression> ::= <term> | <term> "+" <expression>
<term>       ::= <factor> | <factor> "*" <term>
<factor>     ::= <constant> | <variable> | "(" <expression> ")"
<variable>   ::= "x" | "y" | "z"
<constant>   ::= <digit> | <digit> <constant>
<digit>      ::= "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9"
```

EBNF:

```
expression = term , [ "+" , expression ];
term       = factor , [ "*" , term ];
factor     = constant | variable | "(" , expression , ")";
variable   = "x" | "y" | "z";
constant   = digit , { digit };
digit      = "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9";
```

ABNF:

```
expression = term ["+" expression]
term       = factor ["*" term]
factor     = constant / variable / "(" expression ")"
variable   = "x" / "y" / "z"
constant   = 1*digit
DIGIT      = "0" / "1" / "2" / "3" / "4" / "5" / "6" / "7" / "8" / "9"
```



Yacc/Lex
Bison

Lenguaje C

Intro

Referencias oficiales

Estándares

1. ANSI C, 1983-1989.
   - ISO/IEC 9899:1990.
   - Conocido como: C89, C90.
2. C11, ISO/IEC 9899:2011
3. C17 o C18, ISO/IEC 9899:2018
4. C23 , ISO/IEC 9899:2024

Veamos algunos drafts:

C17: https://files.lhmouse.com/standards/ISO%20C%20N2176.pdf
C23: https://open-std.org/JTC1/SC22/WG14/www/docs/n3467.pdf

Por ejemplo, si buscamos “long int” en la página 23.

Lenguaje C

Intro

Etapas del proceso de creación

1. Crear programa
2. Compilar
3. Ligar
   i. Estático
   ii. Dinámico
4. Ejecutable
   i. Cargador

   Sistema Operativo, Máquina Virtual

Lenguajes Compilados

vs

Lenguajes Interpretados

Lenguaje C

Intro

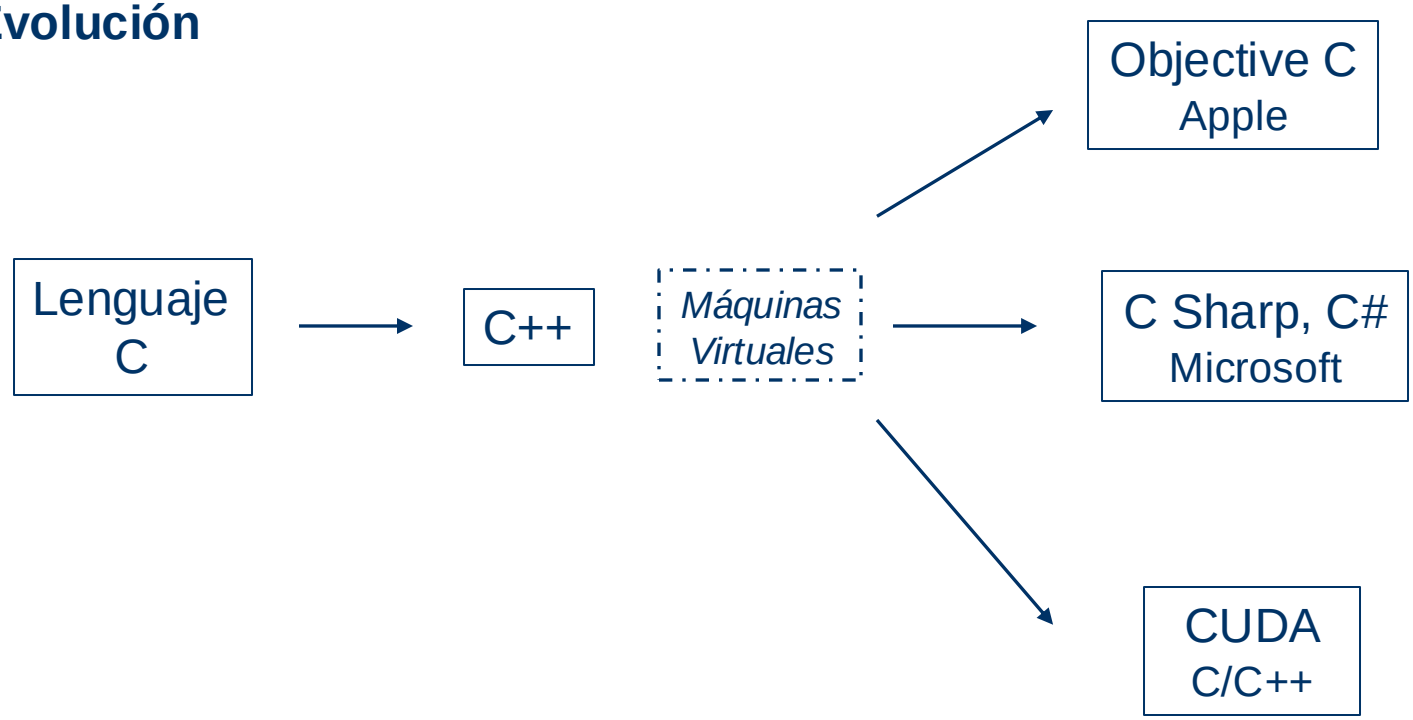
Buenas practices de programación

1. Programación estructurada
   i. Usar estructuras de control de flujo: if-else, while, etc.
   ii. No usar Goto (programación espagueti).
2. Programación modular
   i. Dividir el programa en “pedazos” de código: funciones.
   ii. Encapsular la “funcionalidad”.
   iii. Privilegiar la comprensión sobre la eficiencia.
3. Documentar el código
   i. No solo el encabezado
   ii. Prototipo de la función
      - Parametros
      - Valores de retorno
      - Semántica

Lenguaje C

Intro

Evolución



Lenguaje C++

Conceptos básicos

Sintaxis

Semántica

Lenguaje C++

Intro

Referencias oficiales

Estándares

1. C++23.
2. C++20
3. C++17
4. C++14
5. C++11
6. C++03
7. C++98

URLs

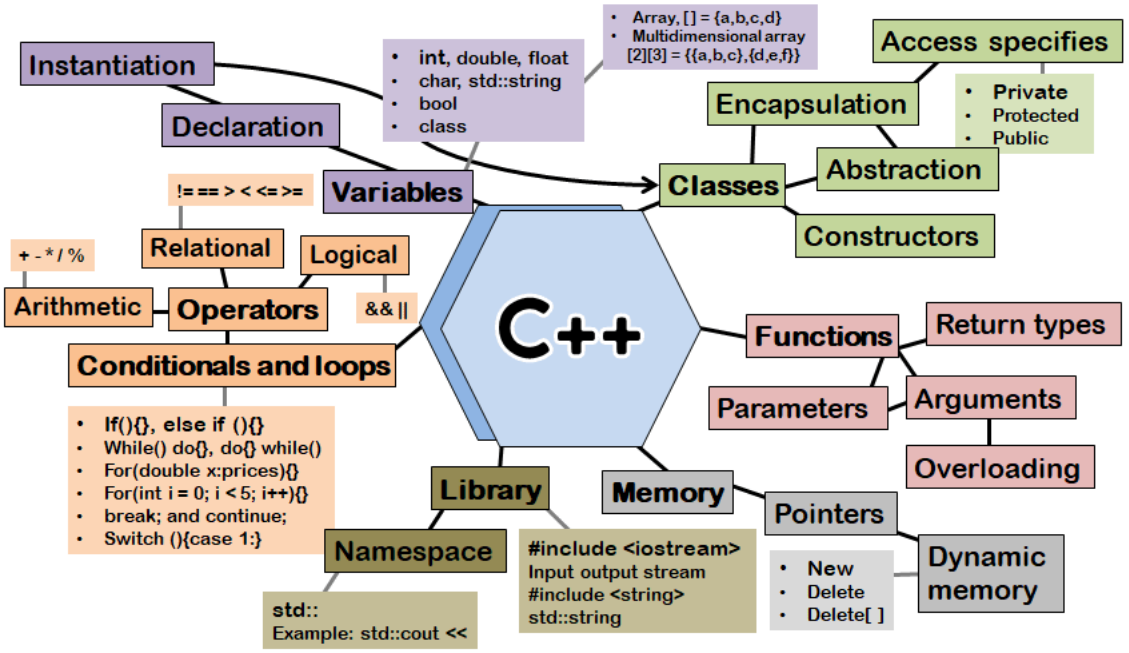
1. https://en.cppreference.com/w/
2. Varios drafts:
   i. https://www.open-std.org/
   ii. Veamos uno:

https://www.open-std.org/JTC1/SC22/WG21/docs/papers/2023/n4950.pdf

Lenguaje C++

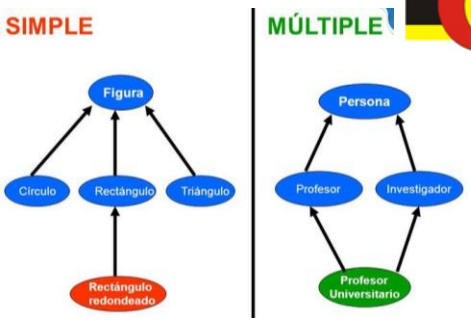
Intro

C++ = Lenguaje C + Conceptos de OO



Conceptos

OO



Concepto	Definición	Especificación
Clase	Plantilla para crear objetos que encapsula atributos (variables) y métodos (funciones). Una misma función puede tener varios prototipos.	Polimorfismo
Encapsulamiento	Oculta los detalles internos de la clase y controla el acceso a los datos mediante modificadores de acceso.	private, protected, public
Herencia	Creación de objetos a partir de otras clases base.	Múltiple
Objeto	Instancia de una clase.	Estáticos y Dinámicos: New
Sobrecarga de operadores	Capacidad de reutilizar los operadores básicos para representar otras operaciones.	
Método	Función ejecutada al recibir un mensaje.	Constructor Destructor Inicialización
Abstracción	Permite crear *interfaces* de los objetos, completas o no.	Método virtual

Lenguaje C++

Intro

Cuando usar OO

Ventajas	Desventajas
Modularidad	Mayor consumo de memoria
Reutilización de código	Baja en el rendimiento
Mantenimiento y Escalabilidad Intercambio de objetos y Fábrica	Curva de aprendizaje
Encapsulamiento y Seguridad	Complejidad en el diseño
Flexibilidad, polimorfismo	
Usado en GUIs	Inadecuado para sistemas de Tiempo Real y Embebidos

Simula (1967)
Smalltalk (1972)
Objective C (1980)
Java (1983)
C++ (1985)
C# (1999)

Semántica

Teoría

Resumen

Semántica

Intro

Tipos

- Semántica Operacional
  - Adecuada para lenguajes imperativos.
- Semántica Axiomática
  - Adecuada para lenguajes lógicos.
- Semántica Denotacional
  - Adecuada para lenguajes funcionales.
  - También se adaptó a los OO.

Semántica

Intro

Formalización del paralelismo y otros aspectos

La mayoría basado en: Procesos + Canales (mensajes)

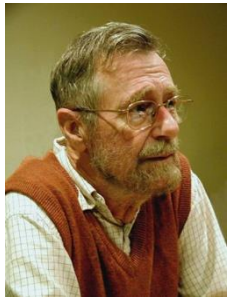
- CCS (Calculus of Communicating Systems, Robin Milner 1980)
  - Adecuada para lenguajes imperativos.
- CSP (Communicating Sequential Processes , Tony Hoare, 1978)
  - Adecuada para lenguajes lógicos.
- Cálculo Lambda (Alonzo Church, 1930-1940)
  - Adecuado para lenguajes funcionales.

Esto dio paso al surgimiento de los Métodos Formales.

- Muchos formalismos que modelan sistemas con matemáticas.
- UNITY, Método B, Notación Z, Lotos, etc.
- Especificación formal

Semántica

Intro



Dijkstra
https://en.wikipedia.org/wiki/On_the_Cruelty_of_Really_Teaching_Computer_Science

https://www.cs.utexas.edu/~EWD/

https://www.cs.utexas.edu/~EWD/transcriptions/EWD10xx/EWD1036.html
https://www.cs.utexas.edu/~EWD/misc/vanVlissingenEntrevista.html

Debate

Métodos Formales

Vs

Ingeniería de Software

Ampliamente usados en sistemas críticos

Ampliamente usado en sistemas informáticos

Scrum, Agile, RUP

Semántica

Intro

Referencias

1. https://www.fmeurope.org/
2. https://formalmethods.fandom.com/
3. https://ntrs.nasa.gov/search?q=Formal%20Methods

Resumen

Conceptos a recordar

Lista

Resumen

A recordar

Conceptos

Los fundamentales:

1. Paradigmas de programación: imperativo, funcional, lógico, etc.
2. Tipos de lenguajes: compilados e interpretados.
3. Entornos de desarrollo: IDE, sdk, toolkit.
4. Ciclo de vida del desarrollo del software
   - Etapas, DevOps, DevSecOps.
   - Pruebas estáticas, dinámicas.
5. Sintaxis: BNF, estándares, etc,
6. Semánticas: operacional, denotacional, axiomática, etc.

The end

Contacto

Raúl Acosta Bermejo

http:www.cic.ipn.mx
http://www.ciseg.cic.ipn.mx/

racostab@ipn.mx
racosta@cic.ipn.mx

57-29-60-00
Ext. 56652