

Introducción a la Programación en C

Ivan Cruz Aceves

ivan.cruz@cimat.mx

Centro de Investigación en Matemáticas, A.C. (CIMAT)

Agosto del 2019

Contenido

Generalidades

Sintaxis

Estructuras de Control

Palabras Clave

Compiladores

Entornos de Desarrollo Integrado (IDE's)

Generalidades

- ▶ Lenguaje desarrollado en 1972 por Dennis Ritchie en los Laboratorios Bell.

Generalidades

- ▶ Lenguaje desarrollado en 1972 por Dennis Ritchie en los Laboratorios Bell.
- ▶ Se enfoca principalmente para generar sistemas operativos, compiladores o aplicaciones científicas, pero existen infinidad de aplicaciones creadas en este lenguaje.






















































































Generalidades

- ▶ Lenguaje desarrollado en 1972 por Dennis Ritchie en los Laboratorios Bell.
- ▶ Se enfoca principalmente para generar sistemas operativos, compiladores o aplicaciones científicas, pero existen infinidad de aplicaciones creadas en este lenguaje.
- ▶ Lenguaje de Alto nivel. (Maquina, ensamblador)

Generalidades

- ▶ Lenguaje desarrollado en 1972 por Dennis Ritchie en los Laboratorios Bell.
- ▶ Se enfoca principalmente para generar sistemas operativos, compiladores o aplicaciones científicas, pero existen infinidad de aplicaciones creadas en este lenguaje.
- ▶ Lenguaje de Alto nivel. (Maquina, ensamblador)
- ▶ Biblioteca estándar.

Ranking IEEE

Language Rank	Types	Spectrum Ranking	Language Rank	Types	Trending Ranking	Language Rank	Types	Jobs Ranking
1. Python	  	100.0	1. Python	  	100.0	1. Python	  	100.0
2. C++	  	99.7	2. C++	  	96.7	2. Java	  	99.2
3. Java	  	97.5	3. Java	  	94.6	3. C	  	98.8
4. C	  	96.7	4. C	  	93.7	4. C++	  	94.6
5. C#	  	89.4	5. Go	 	85.5	5. C#	  	86.2
6. PHP		84.9	6. JavaScript	 	80.8	6. JavaScript	 	85.7
7. R		82.9	7. PHP		79.9	7. Assembly	 	83.4
8. JavaScript	 	82.6	8. Scala	 	78.6	8. PHP		83.1
9. Go	 	76.4	9. Ruby	 	77.2	9. HTML		81.3
10. Assembly		74.1	10. HTML		75.5	10. Scala	 	76.5
11. Matlab		72.8	11. Assembly	 	75.0	11. Shell		76.3
12. Scala	 	72.1	12. C#	  	74.0	12. Ruby	 	75.5
13. Ruby	 	71.4	13. Shell		72.6	13. Matlab		68.2
14. HTML		71.2	14. R		72.0	14. R		67.5

https://spectrum.ieee.org/ns/IEEE_TPL_2018/index/2018

Ranking TIOBE

Aug 2019	Aug 2018	Change	Programming Language	Ratings	Change
1	1		Java	16.028%	-0.85%
2	2		C	15.154%	+0.19%
3	4	▲	Python	10.020%	+3.03%
4	3	▼	C++	6.057%	-1.41%
5	6	▲	C#	3.842%	+0.30%
6	5	▼	Visual Basic .NET	3.695%	-1.07%
7	8	▲	JavaScript	2.258%	-0.15%
8	7	▼	PHP	2.075%	-0.85%
9	14	▲▲	Objective-C	1.690%	+0.33%
10	9	▼	SQL	1.625%	-0.69%
11	15	▲▲	Ruby	1.316%	+0.13%
12	13	▲	MATLAB	1.274%	-0.09%
13	44	▲▲	Groovy	1.225%	+1.04%
14	12	▼	Delphi/Object Pascal	1.194%	-0.18%

<https://www.tiobe.com/tiobe-index/>

Sintaxis en Lenguaje C

Sintaxis: Conjunto de reglas preestablecidas para lograr que un código de programación pueda ser reconocido por un compilador.

Sintaxis en Lenguaje C

Sintaxis: Conjunto de reglas preestablecidas para lograr que un código de programación pueda ser reconocido por un compilador.

```
main()  
{  
  
}
```

- ▶ Programa más sencillo.
- ▶ Función `main`
- ▶ Parentesis posterior a la función (no importa el número de argumentos).
- ▶ Bloque de sentencias encerrado entre llaves.
- ▶ El Lenguaje C, obliga a declarar las variables antes de utilizarlas.
- ▶ Inicialización de variables.

Sintaxis en Lenguaje C

```
#include <stdio.h>
main(){printf("Hola");printf("mundo\n");}
```

- ▶ Sintaxis valida, no muy legible
- ▶ Es común realizar una sentencia por linea, aunque no obligatorio.
- ▶ Cada sentencia se delimita por ;
- ▶ `printf()` función estándar.
- ▶ Inclusión de biblioteca *stdio* (standard input/output).
Fichero o header `stdio.h`
- ▶ `Return 0;`, es deseable agregarlo al fin del programa para avisarle al S.O. que la ejecución termino correctamente.

Operadores y precedencia

¿Cuál sería el resultado?

$$fx = 2 * 5 * 5 + 3 * 5 + 7$$

Operadores y precedencia

¿Cuál sería el resultado?

`fx = 2 * 5 * 5 + 3 * 5 + 7`

<code>()</code>	Izq a Der
-----------------	-----------

<code>* / %</code>	Izq a Der
--------------------	-----------

<code>+ -</code>	Izq a Der
------------------	-----------

<code>< <= > >=</code>	Izq a Der
------------------------------------	-----------

<code>== !=</code>	Izq a Der
--------------------	-----------

72

Operadores y precedencia

Adivinando un número

Operadores y precedencia

Adivinando un número

- ▶ Piensa/ingresa un número (x)
- ▶ Multiplicar por 5
- ▶ Sumar 1
- ▶ Multiplicar por 2
- ▶ Restar 12
- ▶ Dividir por 10
- ▶ Sumar 1

Operadores y precedencia

Adivinando un número

- ▶ Piensa/ingresa un número (x)
- ▶ Multiplicar por 5
- ▶ Sumar 1
- ▶ Multiplicar por 2
- ▶ Restar 12
- ▶ Dividir por 10
- ▶ Sumar 1

```
int dato;  
printf ("Ingresa un numero:");  
scanf ("%d",&dato);
```

Realizar programa en C (1 sentencia, sin parentesis)

Estructuras de control

- ▶ Estructura de selección *if*
- ▶ Estructura de selección *if/else*
- ▶ Estructura de selección múltiple *Switch*
- ▶ Estructura de repetición *do/while*
- ▶ Estructura de repetición *for*

Estructura de selección if/ operador condicional

```
if(condicion)
printf("Escritura\n");
```

```
if(condicion)
printf("Escritura\n");
else
printf("Lectura\n");
```

```
condicion?accion 1: accion 2
printf("%s\n",condicion?accion 1: accion 2);
```

```
if(){...} else
if(){...} else
if(){...}
else
```

Estructura de selección if/ operador condicional

```
if(condicion)
printf("Escritura\n");
```

```
if(condicion)
printf("Escritura\n");
else
printf("Lectura\n");
```

```
condicion?accion 1: accion 2
printf("%s\n",condicion?accion 1: accion 2);
```

```
if(){...} else
if(){...} else
if(){...}
else
```

Ejercicio: Imprimir en orden descendente 3 números ingresados por teclado.

Estructura de selección múltiple Switch

```
switch(variable)
{
case comparacion1:
accion...
break;
case comparacion2:
accion...
break;
default:
accion...
break;
}
```

Estructura de repetición do/while

```
do  
{  
sentencias  
...  
}while(condicion);
```

Estructura de repetición do/while

```
do
{
sentencias
...
}while(condicion);

while (condicion)
{
sentencias
...
}
```

Estructura de repetición do/while

```
do
{
sentencias
...
}while(condicion);

while (condicion)
{
sentencias
...
}
```

Realizar programa en C que a partir de un número por consola, regrese sus dígitos en reversa.

Estructura de repetición do/while

```
do
{
sentencias
...
}while(condicion);

while (condicion)
{
sentencias
...
}
```

Realizar programa en C que a partir de un número por consola, regrese sus dígitos en reversa.

Tarea: Determinar si dicho número es palíndromo

Ejercicio previo en C

```
#include <stdio.h>
int main()
{
    int m,d,n=0;
    printf ("Ingresa un numero:");
    scanf("%d",&m);
    while(m > 0)
    {
        d= m%10;
        m /=10;
        n = 10*n+d;
    }
    printf("\nEl reverso es: %d",n);
    return 0;
}
```

Estructura de repetición for

```
for(variable = valor inicial;condicion;incremento)
{
    sentencias
    ...
}
```

Estructura de repetición for

```
for(variable = valor inicial;condicion;incremento)
{
    sentencias
    ...
}
```

Ejercicio: Ingresar por teclado un numero binario (entero), y mostrar su conversión a número decimal.

Estructura de repetición for

```
for(variable = valor inicial;condicion;incremento)
{
    sentencias
    ...
}
```

Ejercicio: Ingresar por teclado un numero binario (entero), y mostrar su conversión a número decimal.

Tarea: Del número binario previo mostrar su conversión al sistema octal y hexadecimal.

Palabras Clave

```
auto break continue char const case default do  
double else enum extern float for goto if  
int long register return short signed sizeof static  
struct switch typedef union unsigned void volatile while
```

Palabras Clave

auto break continue char const case default do
double else enum extern float for goto if
int long register return short signed sizeof static
struct switch typedef union unsigned void volatile while

auto: no suele usarse, declara que una variable es local a una subrutina. Por defecto toda variable local es auto. **goto**: Salto incondicional a otra parte del programa (etiqueta). **extern**: modificador para variables globales entre módulos diferentes. **register**: para almacenar variables en el registro del computador. S.O. **volatile**: Indica al compilador que debe evitar realizar optimización sobre la variable. Es muy empleado en mapeo de hardware. S.O.

Para mañana

!!! Compiladores e IDE's !!!

El Compilador

Compilador: Es el encargado de traducir el código fuente a lenguaje máquina o código binario.

El Compilador

Compilador: Es el encargado de traducir el código fuente a lenguaje máquina o código binario.

El proceso de compilación consta de 4 etapas, aunque el GCC las realiza todas en un sólo paso.

El Compilador

Compilador: Es el encargado de traducir el código fuente a lenguaje máquina o código binario.

El proceso de compilación consta de 4 etapas, aunque el GCC las realiza todas en un sólo paso. Etapas de Compilación:

Preprocesado: Interpreta las directivas del preprocesador. Por otra parte, es legible la sustitución de las constantes `#define` por su valor numérico.

El Compilador

Compilador: Es el encargado de traducir el código fuente a lenguaje máquina o código binario.

El proceso de compilación consta de 4 etapas, aunque el GCC las realiza todas en un sólo paso. Etapas de Compilación:

Preprocesado: Interpreta las directivas del preprocesador. Por otra parte, es legible la sustitución de las constantes `#define` por su valor numérico.

Compilación: Transforma el código C en lenguaje ensamblador de acuerdo al procesador de la computadora.

El Compilador

Compilador: Es el encargado de traducir el código fuente a lenguaje máquina o código binario.

El proceso de compilación consta de 4 etapas, aunque el GCC las realiza todas en un sólo paso. Etapas de Compilación:

Preprocesado: Interpreta las directivas del preprocesador. Por otra parte, es legible la sustitución de las constantes `#define` por su valor numérico.

Compilación: Transforma el código C en lenguaje ensamblador de acuerdo al procesador de la computadora.

Ensamblado: Transforma el programa escrito en lenguaje ensamblador de la etapa previa a código objeto.

El Compilador

Compilador: Es el encargado de traducir el código fuente a lenguaje máquina o código binario.

El proceso de compilación consta de 4 etapas, aunque el GCC las realiza todas en un sólo paso. Etapas de Compilación:

Preprocesado: Interpreta las directivas del preprocesador. Por otra parte, es legible la sustitución de las constantes `#define` por su valor numérico.

Compilación: Transforma el código C en lenguaje ensamblador de acuerdo al procesador de la computadora.

Ensamblado: Transforma el programa escrito en lenguaje ensamblador de la etapa previa a código objeto.

Enlazado: Incorpora el código binario de las bibliotecas existentes (`printf`, `scanf`, etc..) en el sistema con el ejecutable del código objeto de nuestro programa.

El Compilador

Algunos compiladores de C:

- ▶ GNU Compiler Collection (GCC)
- ▶ MinGW (GCC para Windows, minimalist GNU for Windows) (Debido a que C fue diseñado para sistemas UNIX, los compiladores para Windows no estan tan extendidos)
- ▶ Borland Turbo C (Descontinuado)
- ▶ Visual C++ (mismo nombre IDE y Compilador)
- ▶ Portable C Compiler (Sistemas BSD)
- ▶ Intel C++ Compiler
- ▶ CLang / LLVM

Entorno de Desarrollo Integrado (IDE)

- ▶ IDE: Software que contiene varias herramientas que ayudan al desarrollo de un programa informático.

Entorno de Desarrollo Integrado (IDE)

- ▶ IDE: Software que contiene varias herramientas que ayudan al desarrollo de un programa informático.
- ▶ Dichas herramientas pueden ser:
 - ▶ Editor de código fuente.
 - ▶ Compilador
 - ▶ Depurador
 - ▶ Explorador de archivos
 - ▶ Control de versiones

Entorno de Desarrollo Integrado (IDE)

- ▶ Eclipse, multiplataforma
<https://www.eclipse.org/downloads/packages/release/helios/sr2/eclipse-ide-cc-developers>
- ▶ DEV-C++ <http://orwelldevcpp.blogspot.com/>
- ▶ Code::Blocks, multiplataforma
<http://www.codeblocks.org/>
- ▶ CLion, multiplataforma, paga
<https://www.jetbrains.com/clion/>
- ▶ CodeLite, multiplataforma <https://codelite.org/>
- ▶ Geany, multiplataforma <https://www.geany.org/>
- ▶ Microsoft Visual C++
- ▶ Visual Studio C++
- ▶ Xcode, MacOS, GCC
- ▶ Notepad++, sublime text, atom, Netbeans

Trabajando con GNU GCC

`gcc`: Compilador para programas escritos en Lenguaje C.

`g++`: Compilador para programas escritos en Lenguaje C++, aunque también puede compilar en Lenguaje C.

Trabajando con GNU GCC

`gcc`: Compilador para programas escritos en Lenguaje C.

`g++`: Compilador para programas escritos en Lenguaje C++, aunque también puede compilar en Lenguaje C.

- ▶ MinGW, dar de alta GCC en PATH de Windows
 - ▶ Propiedades del Sistema
 - ▶ Variables de entorno
 - ▶ PATH, modificar
 - ▶ Agregar ruta de GCC, separado por;

Opciones más comunes en GCC

- ▶ `gcc --version`
- ▶ `gcc archivo.c` (genera archivo ejecutable)
- ▶ `g++ archivo.c`
- ▶ `gcc archivo.c -o archivo.exe` (nombre archivo de salida)
- ▶ `g++ archivo.c -o archivo.exe`
- ▶ `gcc -c archivo.c` (genera código objeto, no realiza enlazado)
- ▶ `gcc -Wall archivo.c -o archivo.exe` (Mostrar warnings)
- ▶ `gcc archivo.c -o archivo.exe -g` (incluye al exe información para rastrear errores mediante algún depurador del tipo GDB(GNU Debugger))

Compilado en 4 etapas en GCC

- ▶ Preprocesado
- ▶ `gcc -E ejemplo.c > ejem.pp`

Compilado en 4 etapas en GCC

- ▶ Preprocesado
- ▶ `gcc -E ejemplo.c > ejem.pp`
- ▶ Compilación
- ▶ `gcc -S ejemplo.c` (entrega archivo ejemplo.s)

Compilado en 4 etapas en GCC

- ▶ Preprocesado
- ▶ `gcc -E ejemplo.c > ejem.pp`
- ▶ Compilación
- ▶ `gcc -S ejemplo.c (entrega archivo ejemplo.s)`
- ▶ Ensamblado siguiendo secuencia (archivo.o)
- ▶ `as -o ejemplo.o ejemplo.s`
- ▶ Ensamblado en un sólo paso
- ▶ `gcc -c archivo.c`

Compilado en 4 etapas en GCC

- ▶ Preprocesado
- ▶ `gcc -E ejemplo.c > ejem.pp`
- ▶ Compilación
- ▶ `gcc -S ejemplo.c (entrega archivo ejemplo.s)`
- ▶ Ensamblado siguiendo secuencia (archivo.o)
- ▶ `as -o ejemplo.o ejemplo.s`
- ▶ Ensamblado en un sólo paso
- ▶ `gcc -c archivo.c`
- ▶ Enlazado
- ▶ `ld -o ejemplo.exe ejemplo.o (falta agregar más referencias)`

Compilado en 2 fases y tipos de enlace

- ▶ Generación de código objeto (archivo.o)
- ▶ `gcc -c ejemplo.c`
- ▶ Enlazado
- ▶ `gcc -o final.exe ejemplo.o`
- ▶ varios códigos objeto
- ▶ `gcc -o final.exe ej1.o ej2.o ej3.o`

Compilado en 2 fases y tipos de enlace

- ▶ Generación de código objeto (archivo.o)
- ▶ `gcc -c ejemplo.c`
- ▶ Enlazado
- ▶ `gcc -o final.exe ejemplo.o`
- ▶ varios códigos objeto
- ▶ `gcc -o final.exe ej1.o ej2.o ej3.o`

Enlace dinámico: Crea un ejecutable dependiente de la localización de las bibliotecas. Los enlaces son dinámicos por defecto en C y genera un ejecutable de mucho menor tamaño.

Enlace estático: Crea un programa monolítico al incorporar binarios de funciones al ejecutable del programa. Tamaño mayor de ejecutable.
`gcc -static -o ejecutable.exe codigo.c`