
Centro Nacional de Alta Tecnología
Colaboratorio Nacional de Computación Avanzada


EV-HPC-2017 **Escuela de Veranillo en HPC**

# Introducción a la programación en C

M. Sc. Ricardo Román Brenes - ricardo.roman@ucr.ac.cr

Julio 25, 2017

# Contenidos

# Lenguajes

# Naturales y formales

- Lenguaje: sistema de comunicación con símbolos (palabras, léxico), reglas (sintaxis) y contexto (semántica).
- Lenguaje natural: usado por humanos para propósitos de comunicación.
- Lenguaje formal (artificial): construcción artificial para un propósito específico, como estudio de la lógica, matemática o *programación*.

# Lenguajes de programación

- Lenguaje formal, utilizado para describir las instrucciones o cálculos que pueden ser interpretados por una computadora (o máquina).
- Compuestos por:
  - Léxico: conjunto de expresiones básicas que se pueden formar dado el alfabeto del lenguaje.
  - Reglas sintácticas: descripción de como se pueden combinar los lexemas para generar frases válidas.
  - Reglas semánticas: definen el contexto.

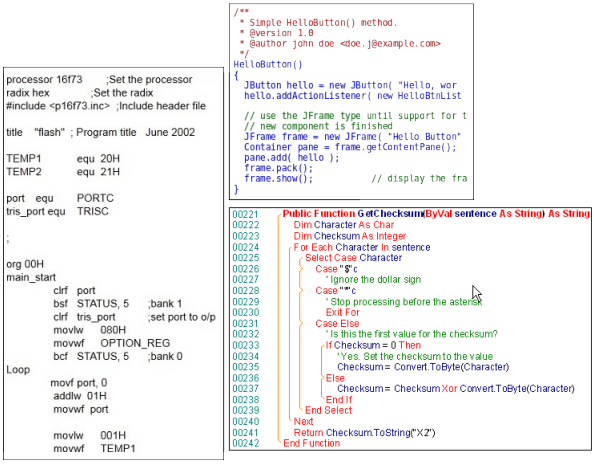
# Software

# Programas y Procesos

- Programa: conjunto de datos e instrucciones utilizados para ejecutar un trabajo o tarea. Escrito en un código fuente.
- Proceso: programa en memoria, instancia de un programa. Ejecutable, código binario.
- Un programa puede ejecutarse varias veces, lo que resulta en diferentes procesos.
- Asociado al proceso, existe:
  - Pila: datos temporales
  - *Heap*: memoria dinámicamente asignada al proceso.
  - Datos: memoria asignada estáticamente.
  - Código: instrucciones del programa.

# Programas y Procesos

## Códigos fuente:



```
processor 16f73      ;Set the processor
radix hex            ;Set the radix
#include <p16f73.inc>  ;Include header file

title   "flash" ; Program title  June 2002

TEMP1       equ  20H
TEMP2       equ  21H

port   equ      PORTC
tris_port equ   TRISC

;

org 00H
main_start
            clrf   port
            bsf   STATUS, 5     ;bank 1
            clrf   tris_port    ;set port to o/p
            movlw     080H
            movwf     OPTION_REG
            bcf   STATUS, 5     ;bank 0
Loop
            movf port, 0
            addlw  01H
            movwf  port

            movlw     001H
            movwf     TEMP1
```

```
/**
 * Simple HelloButton() method.
 * @version 1.0
 * @author john doe <doe.j@example.com>
 */
HelloButton()
{
  JButton hello = new JButton( "Hello, wor
  hello.addActionListener( new HelloBtnList

  // use the JFrame type until support for t
  // new component is finished
  JFrame frame = new JFrame( "Hello Button"
  Container pane = frame.getContentPane();
  pane.add( hello );
  frame.pack();
  frame.show();              // display the fra
}
```

```
00221    Public Function GetChecksum(ByVal sentence As String) As String
00222        Dim Character As Char
00223        Dim Checksum As Integer
00224        For Each Character In sentence
00225            Select Case Character
00226                Case "$"c
00227                    ' Ignore the dollar sign
00228                Case "*"c
00229                    ' Stop processing before the asterisk
00230                    Exit For
00231                Case Else
00232                    ' Is this the first value for the checksum?
00233                    If Checksum = 0 Then
00234                        ' Yes. Set the checksum to the value
00235                        Checksum = Convert.ToByte(Character)
00236                    Else
00237                        Checksum = Checksum Xor Convert.ToByte(Character)
00238                    End If
00239            End Select
00240        Next
00241        Return Checksum.ToString("X2")
00242    End Function
```

Códigos en varios lenguajes.

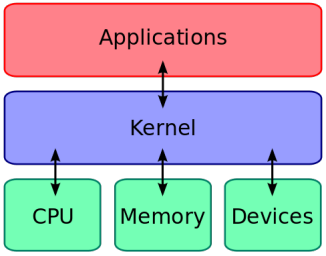
# Sistema Operativo

¿Cómo definir un Sistema Operativo?

- ... es un programa que está en ejecución todo el tiempo.
- ... es un intermediario entre el usuario y el hardware.
- Administrador de recursos:
  - CPU
  - Memoria
  - Almacenamiento
  - Dispositivos de entrada y salida
- Sirve como base para la ejecución de otros programas.

# Sistema Operativo

- Parte o componente principal del S.O.: Kernel
- Entradas del usuario $\rightarrow$ procesos de datos + utilizar dispositivos = resultados.
- Las aplicaciones solicitan recursos al S.O. por medio de llamadas al sistema (*system calls*).




Comunicaciones del Kernel.

# Sistema Operativo

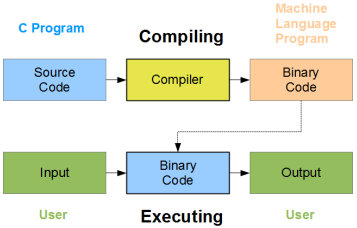
Algunos Sistemas Operativos:

- Windows 3, XP, 2000, 7, 8, 10...
- Mac OS X.
- GNU/Linux: Debian, Fedora, RHEL, CentOS...
- iOS, Android, Symbian...


Sistemas operativos.

# Compiladores

- Un compilador es esencialmente un traductor.
- Convierte un código fuente en un código objeto.
- Por lo general el código fuente es algún le guaje de programación y el código objeto es lenguaje máquina (código binario).




Función del compilador.

# Compiladores

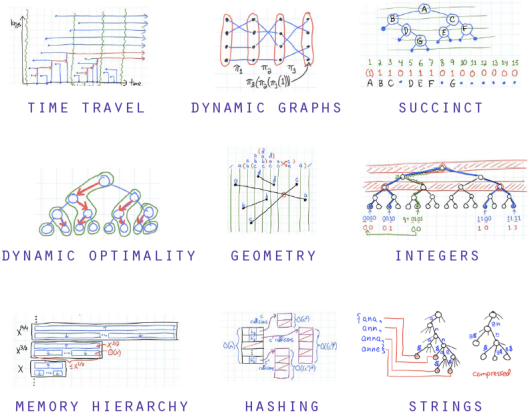
Etapas:


Etapas típicas de un compilador.

# Estructuras de datos

# Estructuras de datos

- La forma en la que se organizan datos.
- Importan para la computadora (eficiencia).
- Diferentes estructuras para diferentes problemas
- Usan memoria/almacenamiento.



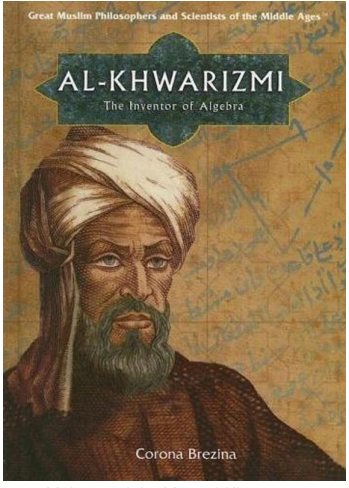

Ejemplos de estructuras de datos.

# Algoritmos

# Algoritmos

- *Trivia*: la palabra algoritmo viene del nombre de un matemático Árabe, quien escribió un tratado sobre matemática, Muhammad ibn Musa al-Khwarizmi.

- Un algoritmo es la formalización de la solución a un problema computable.
- Usan tiempo de procesador.




Muhammad ibn Musa al-Khwarizmi.
http://www.muslimheritage.com/

# Algoritmos

- Un algoritmo es:
  - Conjunto ordenado y finito de instrucciones.
  - Cada instrucción debe ser finita, debe terminar en algún momento.
  - No ambiguo, tan claro como para que un agente sin ingenio lo puede ejecutar.
  - Consistente. La misma entrada debe generar la misma salida.

- Un **buen** algoritmo es:
  - Eficiente.
  - Adaptable y con capacidad de generalización.
  - Elegante, fácil de leer.

# ¿Son todos los programas algoritmos?

# Estructuras de control

# Programación

# ¿Qué es programación?

# Programación

¿Qué es programación (de computadoras)?

- Proceso de automatizar tareas en una computadora.
- Describir los pasos para resolver un problema.
- Uno de los pasos del Ciclo de Vida del Software.
- Darle órdenes a la computadora.
- ...
- **Implementación de un algoritmo**.
  - Formalización detallada, finita y no ambigua de una solución.

# ¿Es necesario aprender C (Java, Fortran, etc) para aprender a programar?

# Programación

¡No!

- Programar es resolver problemas.
- La industria ha confundido la opinión.
- Aprender a programa es independiente de los lenguajes de programación.
- Un buen programador sabe:
  - Algoritmos.
  - Estructuras de datos.
  - Procedimientos.
  - Pensamiento lógico.
  - Innovación.
  - ...

# Programación

**Programa**

- Datos e instrucciones.
- Varias representaciones:
  - **Código fuente** (Java, C, C++, Python, Fortran...)
  - Código intermedio (*bytecode* en Java).
  - Código a máquina (binario)

# Programación

- Escribir código fuente.


- Antes... estructuras de control y datos.

# Datos y tipos de datos

# Tipos de datos

- Los algoritmos operan sobre datos.
- Los datos se representan como **variables** o **constantes**:
  - Las variables pueden cambiar su valor dinámicamente.
  - Las constantes no cambian nunca su valor.
- Ambos tiene un tipo, ya sea explicito (Java, C) o implícito (Python, Javascript):
  - `entero`
  - `real`
  - `booleano`
  - `carácter`
  - `hilera de caracteres`
  - `arreglos`
  - `punteros`

# Identifique las variables y constantes:

```
esPar(){
  escribir “Introduzca un número”
  entero NUMERO
  leer NUMERO
  SI NUMERO>0  ENTONCES
    escribir “Positivo”
  SI NO
    SI NUMERO<0 ENTONCES
      escribir “Negativo”
    SI NO
      escribir “Cero”
    FINSI
  FINSI
}
```

# Procesamiento

# Operaciones y funciones

- Las instrucciones siguen un flujo de ejecución.
- Los datos deben ser transformados con operadores o funciones.
  - **Operadores**: conjunto básico de transformaciones.
  - Cada lenguaje de programación defines sus propios operadores.
  - Algunos operadores comunes:

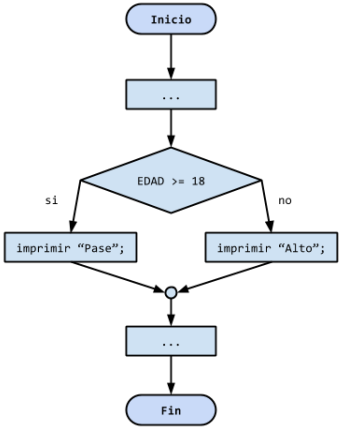
```
() [] -> . ! ~ ++ -- + - * / % ^ < <= > <= == !=
& | && || ?: = += -= *= /= %= &= ^= |=  << >> <<= >>=
```

  - **Funciones**: subrutinas, métodos, procedimientos, etc. Similares a los operadores pero con diferencias sintácticas.
  - En su mayoria definidas por el usuario.
  - En su prototipo o especificación se detallan sus argumentos y valor de retorno.
  - Ej.: `entero sumaDeEnteros(a, b)`

# Estructuras de control

# if/else

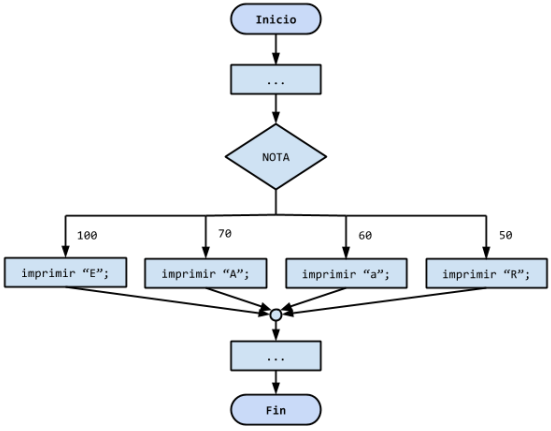
- Prueba lógica.
- Divide el flujo en **dos** posibles caminos.




```
...
  if(EDAD >= 18)
  {
    imprimir "Pase";
  }
  else
  {
    imprimir "Alto";
  }
...
```

# switch/case

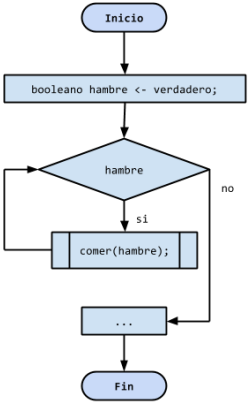
- Prueba lógica.
- Divide el flujo en **varios** posibles caminos.




```
...
  switch(EDAD)
  {
    case 100:
      imprimir "Excelente";
    case 70:
      imprimir "Aprobado";
    case 60:
      imprimir "ampliación";
    case 50:
      imprimir "Reprobado";
  }
...
```

# while

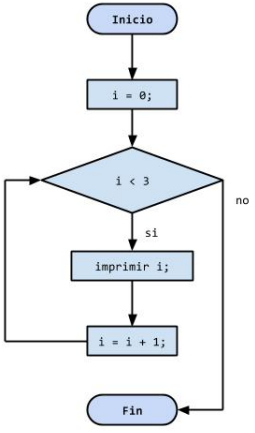
- Prueba lógica.
- Ejecución del cuerpo del ciclo.
- Se itera hasta que la prueba de como resultado **FALSO**.




```
...
  booleano hambre <- verdadero;
  while(hambre)
  {
    comer(hambre);
  }
...
```

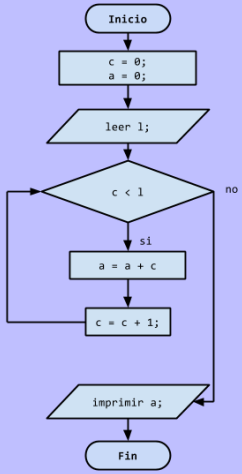
# for

- Se necesita un contador, variable de tipo `entero`.
- Prueba lógica.
- Ejecución del cuerpo del ciclo.
- Actualización del contador.
- Se itera hasta que la prueba de como resultado **FALSO**.




```
...
  entero i = 0;
  for(i; i < 3; i = i + 1)
  {
    imprimir i;
  }
...
```

# ¿Qué hace este programa?




```
s()
{
  c = 0;
  a = 0;
  leer l;
  for(c; c < l; c = c + 1)
  {
    a = a + c;
  }
  imprimir a;
}
```

# Compilador de C

# Compilador

- Toma código fuente en C y lo porta a código a máquina.
- Optimizaciones.
- Diferentes implementaciones del programa traductor:


# GNU

- GNU Compilator Collection.
- Lenguajes: C (gcc), C++ (g++), Objective-C, Objective-C++, Fortran (gfortran), Java (gcj), Ada (GNAT) y Go (gccgo).
- Arquitecturas:

- Alpha
- **ARM**
- AVR
- Blackfin
- Epiphany
- H8/300
- HC12
- **IA-32 (x86)**
- **IA-64 (Itanium)**
- **MIPS**
- **Motorola 68000**
- PA-RISC
- PDP-11
- **PowerPC**
- R8C / M16C / M32C
- **SPARC**
- SPU
- SuperH
- System/390 / zSeries
- VAX
- **x86-64**
- 68HC11
- A29K
- CR16
- C6x
- D30V
- DSP16xx
- ETRAX CRIS
- FR-30
- FR-V
- Intel i960
- IP2000
- M32R
- MCORE
- MIL-STD-1750A
- MMIX
- MN10200
- MN10300
- **Motorola 88000**
- NS32K
- ROMP
- RL78
- Stormy16
- V850
- Xtensa
- Cortus APS3
- ARC
- AVR32
- C166 and C167
- D10V
- EISC
- eSi-RISC
- Hexagon
- LatticeMico32
- LatticeMico8
- MeP
- MicroBlaze
- Motorola 6809
- MSP430
- NEC SX architecture
- Nios II and Nios
- OpenRISC
- PDP-10
- PIC24/dsPIC
- PIC32
- Propeller
- RISC-V
- Saturn (HP48XGCC)
- System/370
- TIGCC
- TriCore
- Z8000
- ZPU

# Intel® C++ Compilers

- Intel® C++ Compilers
- Lenguajes: Fortran ifort, C (icc), C++ (icpc).
- Es software privativo
- Arquitecturas Intel.


Centro Nacional de Alta Tecnología
Colaboratorio Nacional de Computación Avanzada


EV-HPC-2017 **Escuela de Veranillo en HPC**

# Introducción a la programación en C

M. Sc. Ricardo Román Brenes - ricardo.roman@ucr.ac.cr

Julio 25, 2017