Fundamentos de Programación (Curso Propédeutico MIE)

Dr. Enrique Stevens Navarro
Facultad de Ciencias
Universidad Autónoma de San Luis Potosí
Junio 2009

Fundamentos de Programación

- La programación es una herramienta indispensable en el posgrado de Ingeniería Electrónica (tareas, proyectos, tesis, etc.)

- Es por tanto muy necesario reafirmar los conocimientos de programación de los estudiantes en especial bajo un lenguaje de aplicación general como C/C++.

- **OBJETIVO GENERAL**

- Reafirmar los conocimientos del lenguaje C/C++.
- En particular, en los bloques básicos de la programación estructurada.

Contenido del curso

- Unidad 1: Entrada, salida y manejo de variales.
- Unidad 2: Estructuras condicionales.
- Unidad 3: Estructuras iterativas.
- Unidad 4: Funciones
- Unidad 5: Arreglos
- Unidad 6: Ejemplos de Análisis Númerico *

Evaluación y Bibliografía

- Evaluación del curso:
  - Tareas y prácticas 50%
  - Examen #1 25% (Julio 1, 2009)
  - Examen #2 25% (Julio 15, 2009)

- Bibliografia:
  - “Como Programar en C/C++”, H.M. Deiter y P.J. Deitel, Prentice Hall.
  - ……

  - CICTD. http://cictd.uaslp.mx/nueva/cictd/index.html
  - Catálogo UASLP “on line”: http://cictd.uaslp.mx/version/

Tipos de lenguaje de programación

- Lenguajes de Bajo Nivel:
  - Lenguajes totalmente dependientes del hardware de la computadora y por tanto aprovechan al máximo las caracteristicas del mismo.
  - Ejemplos:
    - Lenguaje maquina (0 y 1s)
    - Lenguaje ensamblador

- Lenguajes de Alto Nivel:
  - Lenguajes totalmente independientes del hardware y por tanto altamente portables.
  - Existen de propósito general y especifico con librerias altamente complejas y especializadas.
  - Ejemplos:
    - MATLAB
    - OCTAVE

Tipos de lenguaje de programación (cont)

- Lenguajes de Medio Nivel (controversial):
  - Lenguajes que se encuentran en un punto medio entre los dos anteriores. Pueden acceder a los registros del sistema y direcciones de memoria y a la vez pueden realizar operaciones de alto nivel.
  - Ejemplo:
    - C/C++
- Cada tipo de lenguaje de programación tiene sus ventajas y desventajas. La mayoria de las veces depende del tipo de problema en que estemos trabajando.
- El hecho de reafirmar C/C++ es por ser un lenguaje de programación muy utilizado, robusto y de buen desempeño asi como de aplicación general.

Traducción del Lenguaje de Programación

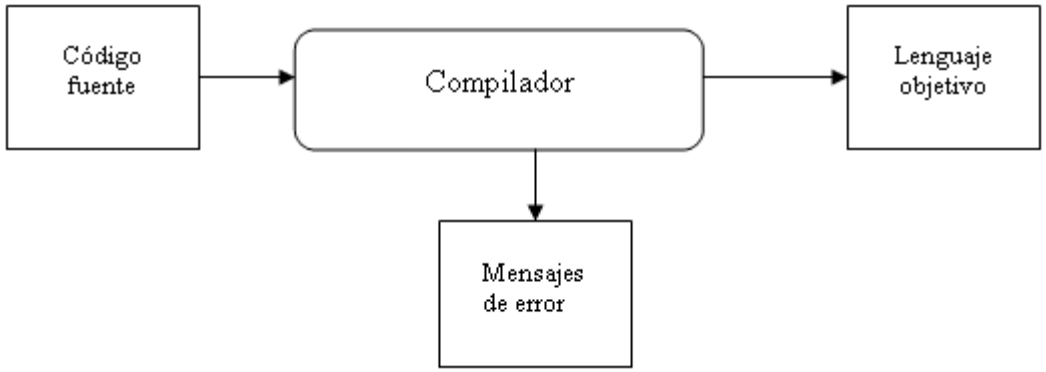
- La computadora NO es políglota….. solo entiende el lenguaje máquina (1 y 0s), así que cualquier programa en un leguaje de programación debe ser traducido al lenguaje máquina.

- Los programas que escribimos en C/C++ son llamados codigo fuente.

- Los compiladores son los encargados de leer el código fuente y traducirlo o convertirlo en otro lenguaje. Los compiladores a la vez pueden detectar errores en el código fuente (ejem: sintaxis).

Traducción del Lenguaje de Programación

- Proceso de compilación:



- Etapas para ejecución de un programa en C/C++:
  - 1.- Editar
  - 2.- Preprocesar
  - 3.- Compilar
  - 4.- Enlazar
  - 5.- Cargar
  - 6.- Ejecutar

Tarea # 1

- 1) Leer Cápitulo 1 del libro “Como programar en C/C++.
- 2) Investigue que es el pseudocódigo y para que se utiliza.
- 3) Realize los siguientes programas en pseudocódigo:
  - 3.1 Un programa que saque el promedio, varianza y desviación estandar de sus calificaciones en licenciatura.
  - 3.2 Un programa que convierta de unidades de potencia (Watts) a unidades de potencia en decibelimetros (dBm) y viceversa.

Seudocódigo

- El seudocódigo es un lenguaje artificial e informal que sirve para el desarrollo de algoritmos de programación.
- El seudocódigo le ayuda al programador “a pensar” la lógica y estrucutra de un programa antes de escribirlo en un lenguaje de programación (C/C++).
- El seudocódigo sirve también para expliar un algoritmo a una audiencia mayor que no necesariamente sabe el lenguaje de programación en que se implemento un algoritmo (muy importante para reportar avances en investigación y desarrollo tecnologico).
- Pero…¿que es un algoritmo?

Algoritmo

- Un algoritmo es un procedimiento para la resolución de un problema.

- El algoritmo indica la solución del problema en terminos:
  - Que acciones deben ejecutarse
  - El orden en que las acciones deben ejecutarse

- Suponga que se tiene el problema:“se poncho una llanta”

- ¿Cual sería el algoritmo para solucionar dicho problema?
  - A) acciones
  - B) orden de las acciones

  A este orden de ejecución de acciones en un programa de computadora se le llama: control de programa.

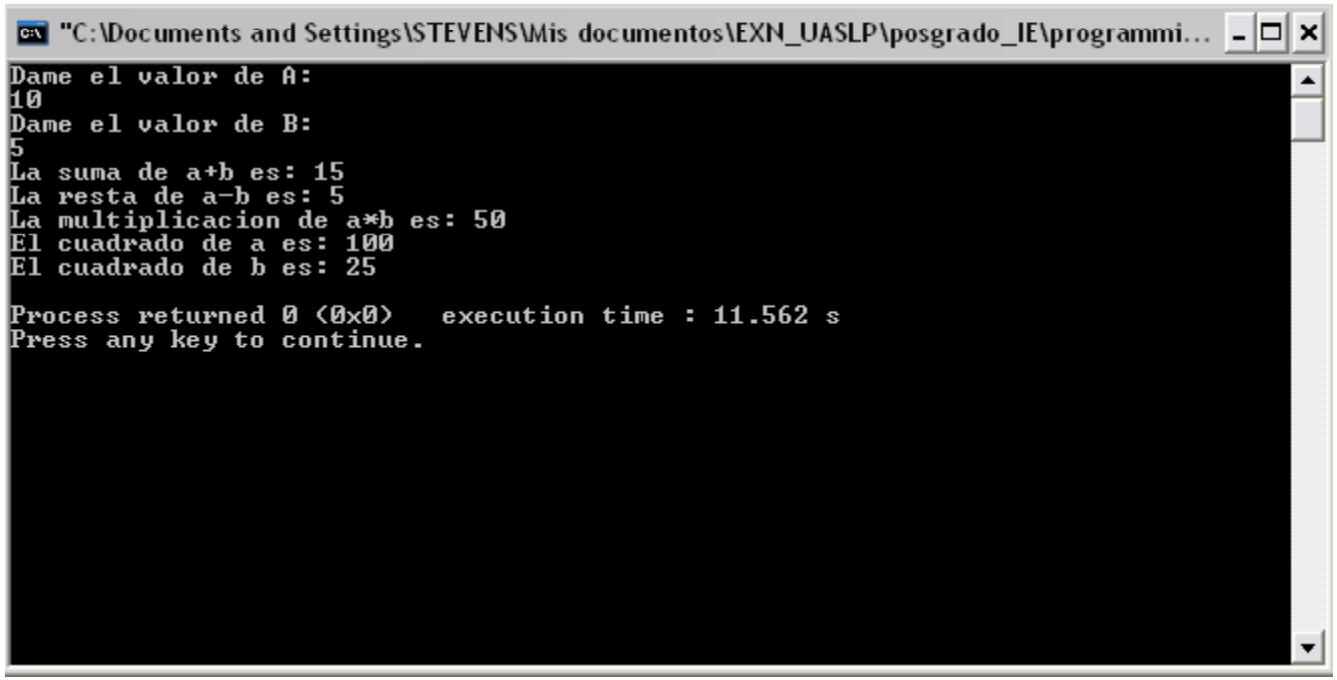
Ejemplo de seudocódigo

```
Programa Aritmética de Primaria

// Este programa realiza la suma, resta,
      multiplicación y cuadrados de dos
      números enteros.

INICIO

// Definición de variables

entero a
entero b
entero suma
entero resta
entero mult
entero a2
entero b2

imprimir “Dame el valor de a:” (a<-)
imprimir “Dame el valor de b:” (b<-)

// Operaciones

suma=a+b
resta=a-b
mult=a*b
a2=a*a
b2=b*b

// Impresión de resultados

imprimir “La suma de a+b es:” (suma)
imprimir “La resta de a-b es:” (resta)
imprimir “La multiplicacion de a*b es:”
      (mult)
imprimir “El cuadrado de a es:” (a2)
Imprimir “El cuadrado de b es:” (b2)

FIN
```

main.c

```c
#include <stdio.h>
#include <stdlib.h>

// Programa Aritmética de Primaria

// Este programa realiza la suma, resta, multiplicación y cuadrados de dos números enteros.

int main()
{
    // Definición de variables

    int a;
    int b;
    int suma;
    int resta;
    int mult;
    int a2;
    int b2;

    printf("Dame el valor de A:\n");
    scanf("%d", &a);
    printf("Dame el valor de B:\n");
    scanf("%d", &b);

    // Operaciones

    suma=a+b;
    resta=a-b;
    mult=a*b;
    a2=a*a;
    b2=b*b;

    // Impresión de resultados

    printf("La suma de a+b es: %d\n", suma);
    printf("La resta de a-b es: %d\n", resta);
    printf("La multiplicacion de a*b es: %d\n", mult);
    printf("El cuadrado de a es: %d\n", a2);
    printf("El cuadrado de b es: %d\n", b2);


    return 0;
}
```

Ejecución del programa en C (consola)



Programa en C

- Todo programa en C es un conjunto de funciones, una de las cuales debera ser la función principal ***main{}***

- Al inicio aparecen las líneas: **#include<……>,** estas líneas del programa son directrices para el preprocesador de C. Le indica al preprocesador que incluya dentro del programa el contenido de algún archivo de cabecera (libreria de funciones) antes del proceso de compilación.

- Algunos de los archivos de cabecera (librerias o .h) mas utilizados son: (Ver Apéndice B del libro de texto)
  - <stdio.h> - Entrada/salida de datos
  - <stdlib.h> - Utilerias generales
  - <math.h> - Matemáticas
  - <time.h> - Fecha y hora

Programa en C

- La cabecera stdio.h incluye las utiles funciones:
  - printf
    - ***printf("Dame el valor de A: \n");***
    - ***printf("La suma de a+b es: %d \t", suma);***
  - scanf
    - ***scanf("%d", &a);***

- El espeficicador de conversión ***%d*** indica que los datos deberan ser un entero.

- Al especificar *&a* se esta haciendo referencia a la dirección de memoria en la cual esta almacenada el valor de la variable ***a***.

Aritmética de primaria en C

operación	operador	en la escuela	en C
Suma	+	a + b	a + b
Resta	-	a - b	a - b
Multiplicación	*	a b	a * b
División	/	a / b, a ÷ b,	a / b
Módulo	%	a mod b	a % b

Precedencia de operadores aritméticos:

1) Expresiones dentro de paréntesis.
2) Operaciones multiplicación, división y módulo.
3) Operaciones suma y resta.

Operadores de igualdad y relaciones en C

operación	operador	en C	significa
=	==	a == b	a es igual b
≠	!=	a != b	a no es igual b
>	>	a > b	a mayor que b
<	<	a < b	a menor que b
≥	>=	a >= b	a mayor igual b
≤	<=	a <= b	a menor igual b

Estructura de Selección if

La estructura if se llama estructura de selección de una sola opción, porque selecciona o ignora una acción.

If en seudocódigo y en C

- Si esto se cumple…. Hago esto…
- Si “tengo dinero”…. “Voy al cine”

Programa número más grande

INICIO

// Definición de variables

entero a=5
entero b=10

si (a > b)
    imprimir “a es el mayor”
Si (a < b)
    imprimir “b es el mayor”
Si (a = b)
    imprimir “a y b son iguales”

FIN

main.c | main.c ×

```c
#include <stdio.h>
#include <stdlib.h>

int main()
{

    int a=5;
    int b=10;

    if (a > b)
        printf("a es el mayor");

    if (a < b)
        printf("b es el mayor");

    if (a == b)
        printf("a y b son iguales");

    return 0;
}
```

Ejercicio en clase

- Para que no se aburra, modifique el seudocódigo del programa de su tarea #1 (de las calificaciones), para que le indique cual fue la materia con mas baja calificación y la materia con mas alta calificación. Si fue mas de una materia el programa debe decir en cuantas materias tuvo la calificación más alta y la más baja.

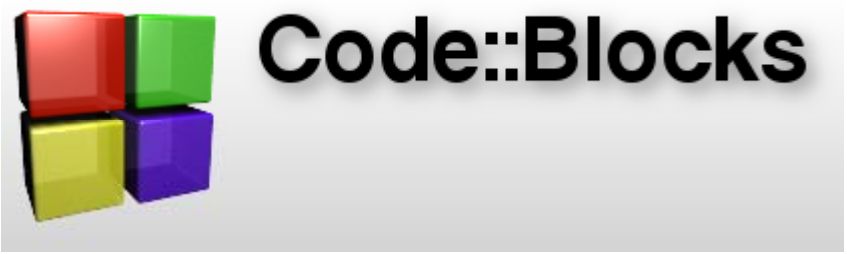
- Favor de entregar el ejercicio junto con la tarea #1.

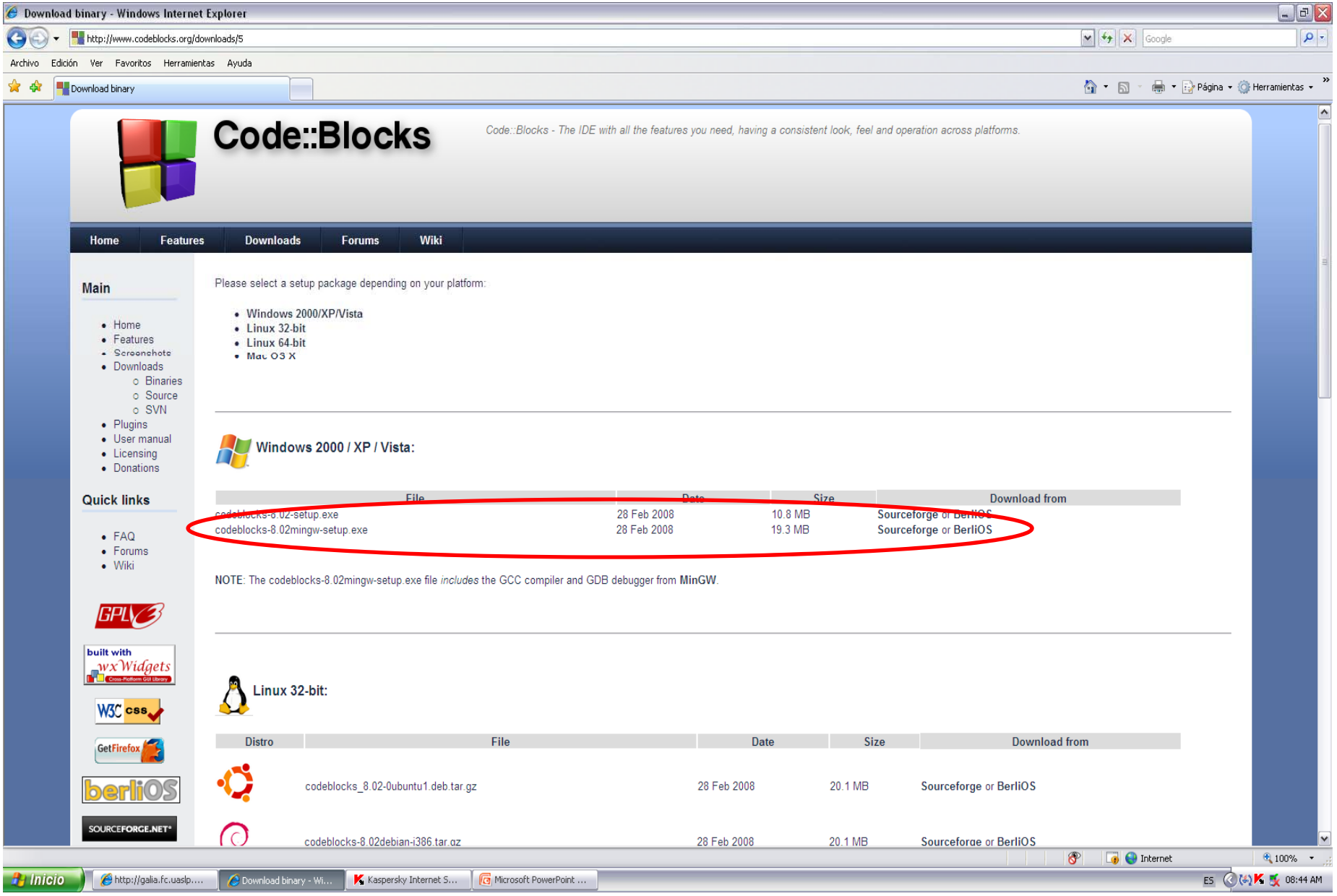
Tarea # 2

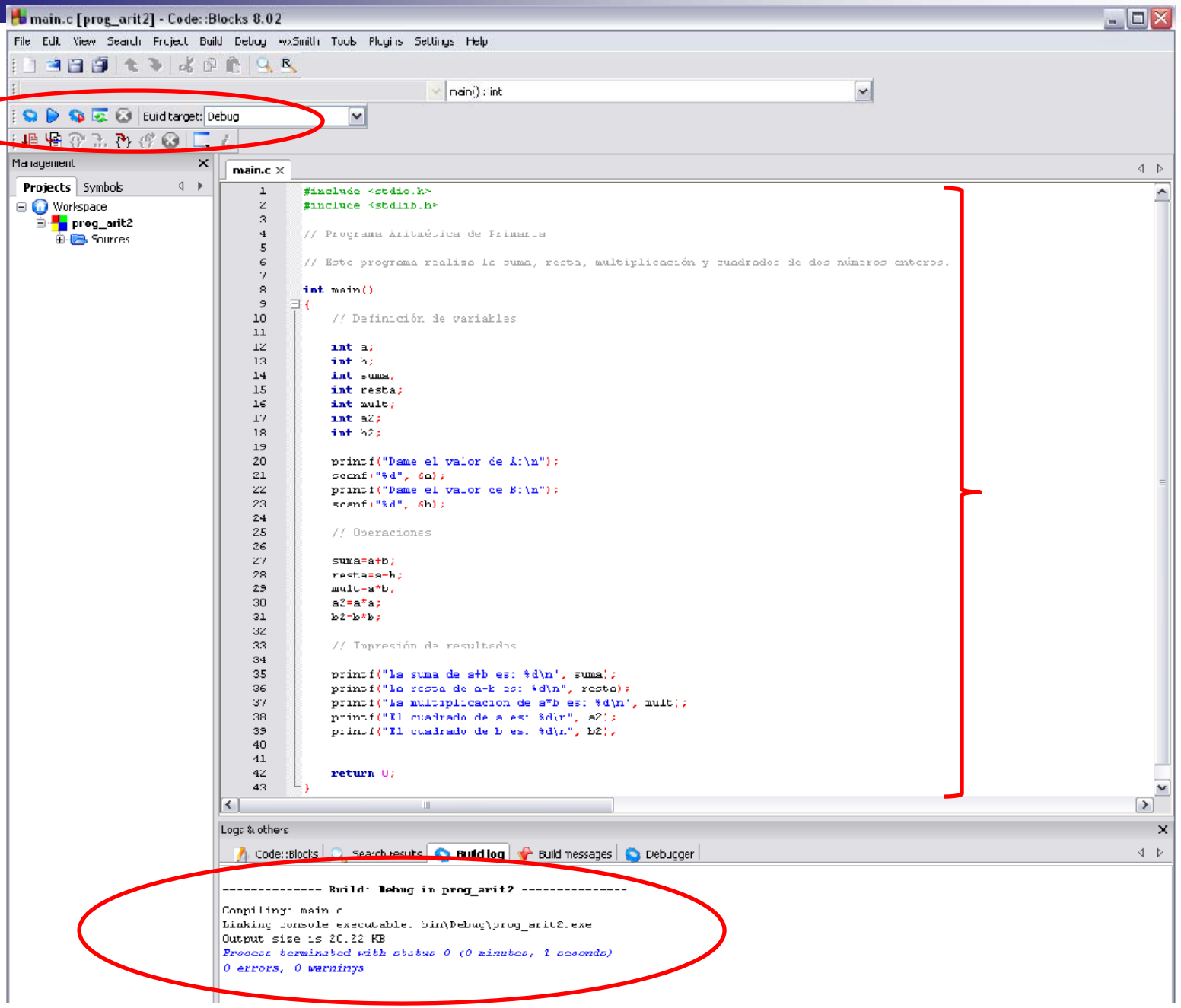
- 1) Leer Cápitulo 2 del libro “Como programar en C/C++”.
- 2) Revise el Apéndice B del mismo libro (librerias).
- 3) Investigue que tipos de variables ofrece el lenguaje C. De cada una indique su rango, espacio que ocupa en memoria y especificador de conversión.
- 4) Realize los siguientes programas en pseudocódigo:
  - 3.1 Un programa que le pida 2 puntos en el plano 2D y que calcule las distancia entre los dos puntos, el punto medio y pendiente de la recta que forman. Ademas imprima la ecuación de la recta que pasa por esos dos puntos.
  - 3.2 Un programa que le pida los coeficientes de una ecuación de segundo grado, la resuelva con formula general y que use el discriminante para saber que tipo de soluciones son (reales o complejas).
  - 3.3 Implemente los programas de las Tareas #1 y #2 en C. Incluya seudocódigo, código fuente y muestre los resultados.

Compiladores

- Existen infinidad de compiladores (muchos libres y otros de licencia).
- Todas las distribuciones de Linux incluyen varios compiladores (cc, gcc, etc.)
- En el caso de Windos tambien hay gran variedad aunque no vienen incluidos en el SO.
- Una opción gratuita para Windows:
  - http://www.codeblocks.org/




Download binary - Windows Internet Explorer
http://www.codeblocks.org/downloads/5
Archivo Edición Ver Favoritos Herramientas Ayuda
Download binary
Página Herramientas
Code::Blocks
Code::Blocks - The IDE with all the features you need, having a consistent look, feel and operation across platforms.
Home Features Downloads Forums Wiki
Main
Home
Features
Screenshots
Downloads
Binaries
Source
SVN
Plugins
User manual
Licensing
Donations
Quick links
FAQ
Forums
Wiki
Please select a setup package depending on your platform:
Windows 2000/XP/Vista
Linux 32-bit
Linux 64-bit
Mac OS X
Windows 2000 / XP / Vista:
File Date Size Download from
codeblocks-8.02-setup.exe 28 Feb 2008 10.8 MB Sourceforge or BerliOS
codeblocks-8.02mingw-setup.exe 28 Feb 2008 19.3 MB Sourceforge or BerliOS
NOTE: The codeblocks-8.02mingw-setup.exe file includes the GCC compiler and GDB debugger from MinGW.
Linux 32-bit:
Distro File Date Size Download from
codeblocks_8.02-0ubuntu1.deb.tar.gz 28 Feb 2008 20.1 MB Sourceforge or BerliOS
codeblocks-8.02debian-i386.tar.gz 28 Feb 2008 20.1 MB Sourceforge or BerliOS
GPLv3
built with wxWidgets
W3C CSS
GetFirefox
berliOS
SOURCEFORGE.NET
Internet
100%
Inicio
http://galia.fc.uaslp....
Download binary - Wi...
Kaspersky Internet S...
Microsoft PowerPoint ...
ES
08:44 AM


main.c [prog_arit2] - Code::Blocks 8.02
File Edit View Search Project Build Debug wxSmith Tools Plugins Settings Help
main() : int
Build target: Debug
Management
Projects Symbols
Workspace
prog_arit2
Sources
main.c
#include <stdio.h>
#include <stdlib.h>
// Programa Aritmética de Primaria
// Este programa realiza la suma, resta, multiplicación y cuadrados de dos números enteros.
int main()
{
// Definición de variables
int a;
int b;
int suma;
int resta;
int mult;
int a2;
int b2;
printf("Dame el valor de A:\n");
scanf("%d", &a);
printf("Dame el valor de B:\n");
scanf("%d", &b);
// Operaciones
suma=a+b;
resta=a-b;
mult=a*b;
a2=a*a;
b2=b*b;
// Impresión de resultados
printf("la suma de a+b es: %d\n", suma);
printf("la resta de a-b es: %d\n", resta);
printf("la multiplicacion de a*b es: %d\n", mult);
printf("El cuadrado de a es: %d\n", a2);
printf("El cuadrado de b es: %d\n", b2);
return 0;
}
Logs & others
Code::Blocks
Search results
Build log
Build messages
Debugger
---------- Build: Debug in prog_arit2 ----------
Compiling: main.c
Linking console executable: bin\Debug\prog_arit2.exe
Output size is 20.22 KB
Process terminated with status 0 (0 minutes, 1 seconds)
0 errors, 0 warnings

Estructuras de Control en Lenguaje C

- Generalmente las instrucciones de un programa son ejecutadas de manera secuencial (ejecución secuencial).
- Sin embargo, existen instrucciones que permiten ejecutar instrucciones que no esten en secuencia dentro del programa (transeferencia de control).
- Caso del legendario comando "goto"
- Fue hasta los 1970's que nacio la programación estructurada cuando se demostro que todos los programas podian ser escritos en terminos de solo **tres** estructuras básicas de control:
  - Estructura de secuencia.
  - Estructura de selección.
  - Estructura de repetición.

Estructuras de Control en Lenguaje C

- Estructura de Secuencia:
- Esta estructura esta en esencia interconstruida a proposito, ya que a menos que se inidique lo contrario la computadora automáticamente ejecuta las intrucciones una después de la otra.

- Estrucutra de Decisión:
- Indica al programa que debe tomar una decisión sobre a donde hacer la transferencia de control (o saltar intrucciones).
- C proporciona **tres** tipos de estructuras de decision:
  - Estructura de decision: **if**
  - Estructura de decision: **if/else**
  - Estructura de decision: **switch**

Estructuras de Control en Lenguaje C

- Estructura if es la estructura de una sola selección.
- Estructura if/else es la estructura de doble selección.
- Estructura switch es la estructura de selección múltiple.

```
Programa ir a cenar (cita)

INICIO

flotante midinero;

imprimir "¿Cuanto dinero tienes:" (midinero<-)

si (midinero > 200)
     imprimir "Invitar a un restaurante"
Else si (midinero > 100)
            imprimir "Invitar a una fonda"
      Else si (midinero > 50)
                imprimir "Invitar a unos tacos (parados)"
            Else imprimir "Hoy no vamos a cenar ☹"
                  imprimir "Ya habra tiempos mejores…"
FIN
```

Ejercicio en clase

- Para que no se aburra, realize el seudocódigo del siguiente programa:

- El programa le debe pedir una calificación (0 – 10).
- Si la calif es 10 debe decir: Excelente
- Si la calif es 9 debe decir: Muy bien
- Si la calif es 8 debe decir: Bien
- Si la calif es 7 debe decir: Regular
- Si la calif es 6 debe decir: Suficiente
- Si la calif es 5 debe decir: Tienes derecho a examen extraordinario.
- Si la calif es menor que 5 debe decir: Tienes derecho a examen a título de suficiencia.

- Utilize la estructura de control if/else.

Ejercicio en clase

- Ahora, realize el seudocódigo del siguiente programa:

- El programa le debe pedir una calificación (0.0 – 10.0).
- Si la calif es 10 debe decir: Excelente
- Si la calif es 9 a 9.9 debe decir: Muy bien
- Si la calif es 8 a 8.9 debe decir: Bien
- Si la calif es 7 a 7.9 debe decir: Regular
- Si la calif es 6 a 6.9 debe decir: Suficiente
- Si la calif es 5 a 5.9 debe decir: Tienes derecho a examen extraordinario.
- Si la calif es menor que 5 debe decir: Tienes derecho a examen a título de suficiencia.

- Utilize nuevamente la estructura de control if/else.

Conexión de Estructuras de Control

- Como se vio en los ejemplos y ejercicios anteriores las estructuras de control pueden ser agregadas unas a otras, es decir, el punto de salida de una con el punto de entrada de la siguiente.

- Ese proceso se llama apilamiento de estructuras de control. (ejem: número mas grande)

- La otra **unica** forma en que se pueden conectar las estructuras de control es que esten unas dentro de otras.

- Ese proceso se llama anidar estructuras de control. (ejem: ir a cenar)

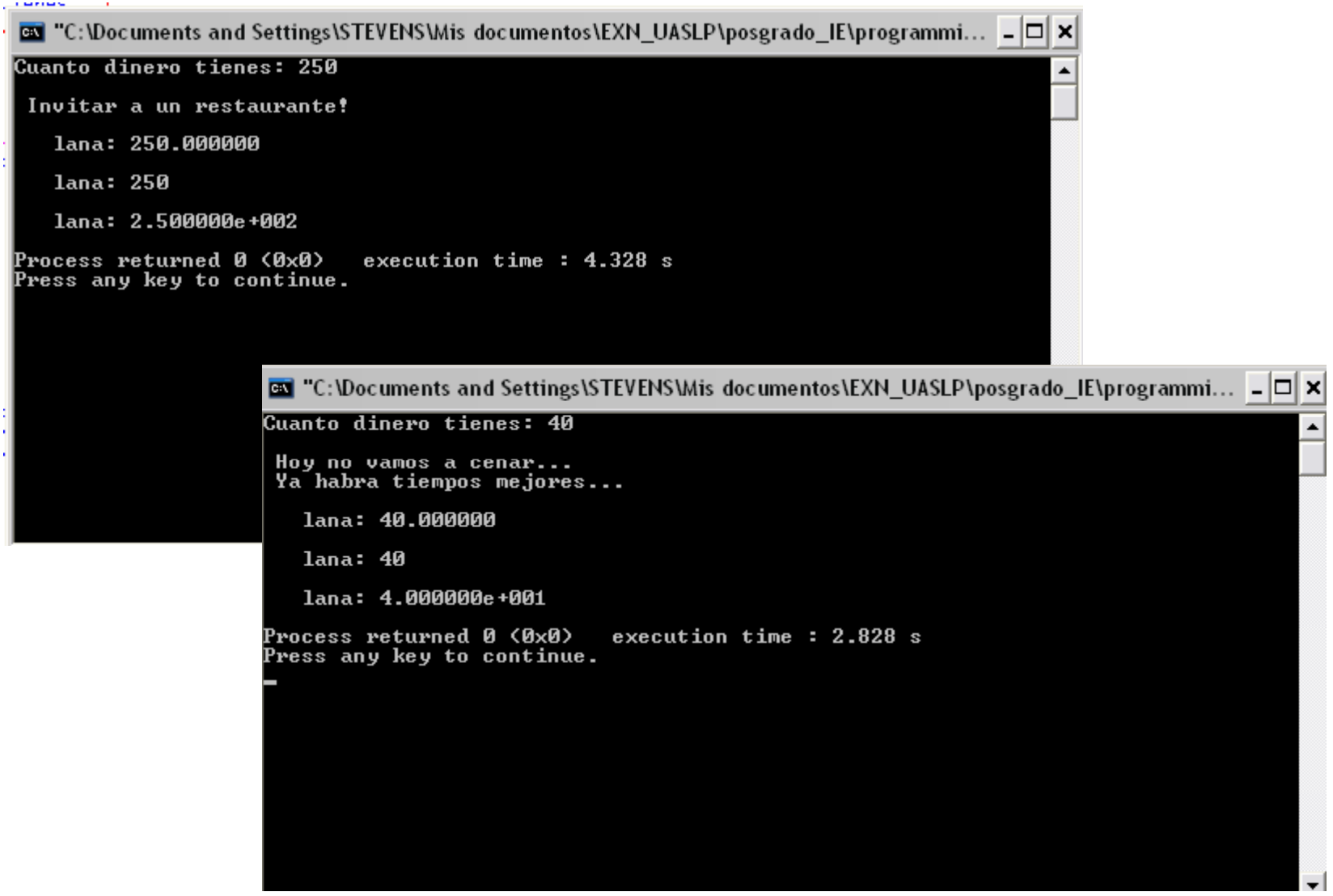
Programa ir a cenar (cita) en C

main.c

```c
#include <stdio.h>
#include <stdlib.h>

// Programa ir a cenar (cita)

int main()
{

    float midinero;

    printf("Cuanto dinero tienes: ");
    scanf("%f", &midinero);

    if (midinero > 200.0)
        printf("\n Invitar a un restaurante!");
    else
        if (midinero > 100.0)
            printf("\n Invitar a una loncheria!");
        else
            if (midinero > 50.0)
                printf("\n Invitar unos tacos (parados)!");
            else
                {
                printf("\n Hoy no vamos a cenar... ");
                printf("\n Ya habra tiempos mejores... ");
                }


    printf("\n\n   lana: %f \n", midinero);
    printf("\n   lana: %g \n", midinero);
    printf("\n   lana: %e \n", midinero);

    return 0;
}

```

- Estructuras if/else anidadas
- Especificadores de conversión para punto flotante.

Ejecución del programa en C



Estructuras de Control en C (Prog. Estructurada)

- En total, el lenguaje C tiene **7** estructuras de control:
  - 1 estructura de secuencia.
  - 3 estructuras de decisión (selección).
    - **if**
    - **if/else**
    - **switch**
  - 3 estructuras de repetición.
    - **while**
    - **do/while**
    - **for**
- Estas estructuras de control pueden ser a su vez:
  - Apiladas.
  - Anidadas.

Estructura de Repetición while

- Nos permite repetir una acción (instrucción o grupo de instrucciones) mientras cierta condición se mantenga verdadera.
- **Mientras** queden cervezas en el refrigerador…
  tomar otra cerveza…

  - Cuerpo de la estructura while.

```
Programa TGIF

INICIO

entero no_cervezas=6;
entero dinero=0;

mientras (no_cervezas != 0)
     imprimir  “Abrir una cerveza…”
     no_cervezas = no_cervezas – 1

si (dinero != 0)
     imprimir “Vamos por mas cerveza…”
else
     imprimir “Vamonos a dormir….”

FIN
```

Tarea # 3

- 1) Leer Cápitulo 3 del libro “Como programar en C/C++”.
- 2) Investigue que es y para que sirve el operador cast.
- 3) Investigue cuales son y para que sirven los operadores incrementales y decrementales en C.
- 4) Realize los siguientes programas en C:
  - 4.1) Modifique su programa en de las calificaciones de manera que funcione para cualquier número de calificaciones (utilize el operador while)
  - 4.2) Implemente los dos ejercicios hechos en clase.
  - 4.3) Modifique el seudocodigo del programa TGIF de manera que le pida una cantidad inicial de dinero y se repita la toma de cervezas hasta que se termine el dinero. El programa debe decir al final el total de cervezas que se tomaron. (Utilize dos while anidados). Asuma que 1 six de cerveza cuesta 50 pesos y que solo puede comprar uno a la vez. Implemente el programa TGIF en C.

Estructuras de Repetición en C

- Existen dos formas de controlar (salir) de la estructura de repetición ***while***:
  - Repetición definida (controlada por contador)
  - Repetición indefinida (controlada por bandera)

- Rep. Definida: antes de que se comienze a ejecutar la estrucutra while se conoce el número de veces que se va a repetir el ciclo.

- Rep. Indefinida: no se conoce el número de veces que se va a repetir el ciclo.

- Un ciclo es un grupo de intrucciones que se ejecutan en forma repetida en tanto se mantenga verdadera la condición de continuación del ciclo.

Programa Promedio Calificaciones

INICIO

entero cont, calif, total, prom;

total=0
cont=1

mientras (cont <= 10)
    imprimir(“Dame la calif:”) (calif<-)
    total=total+calif;
    cont=cont++;

prom=total/10;
Imprimir(“El promedio es: ”, prom)

FIN

- Controlado por contador.

Programa Promedio Calificaciones

INICIO

entero cont, calif, total, prom;

total=0
cont=0

imprimir(“Dame la calif (-1 para terminar): ”) (calif<-)

mientras (calif != -1)
    total=total+calif;
    cont=cont++;
    imprimir(“Dame la calif (-1 para terminar): ”) (calif<-)

prom=total/cont;
Imprimir(“El promedio es: ”, prom)

FIN

- Controlado por bandera.

Ejercicio en clase

- Traduzca del seudocódigo al lenguaje C los dos programas anteriores….
- Tiene solo 8 minutos…

Estructura de Repetición do/while

- La estructura ***do/while*** es similar a la estrucutra ***while.***

- En ***while*** la condición de continuación del ciclo se prueba a principio del ciclo, antes de ejecutarse las intrucciones contenidas en el cuerpo del ***while.***

- La estructura ***do/while*** prueba la condición de continuación del ciclo después de ejecutar el cuerpo del ciclo y por tanto dichas intrucciones por lo menos una vez.

```
while ( condición = verdadera)
    {
    instrucción 1;
    instrucción 2;
    }
```

```
do {
    instrucción 1;
    instrucción 2;
    } while ( condición = verdadera)
```

- No olvidar las { } para agrupar instrucciones dentro de una estructura de control (apiladas y/o anidadas).

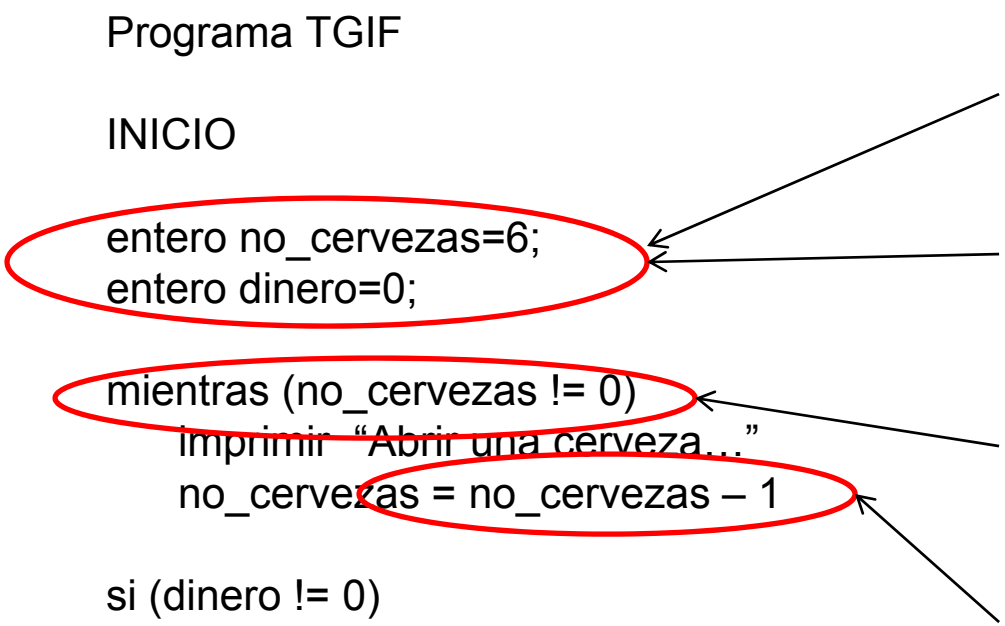
Estructuras de Repetición en C

- En el caso de una estrucutra de repetición controlada por contador, esta require los siguientes elementos:
  - 1) Una variable de control (contador).
  - 2) Un valor inicial de la variable de control.
  - 3) Un incremento o decremento que modificará la variable de control cada vez que termine un ciclo.
  - 4) La condición de continuidad del ciclo.

- En el programa TGIF, ¿puede identificar los 4 elementos anteriores?

Control de Repetición por Contador.

```
Programa TGIF

INICIO

entero no_cervezas=6;
entero dinero=0;

mientras (no_cervezas != 0)
     imprimir “Abrir una cerveza…”
     no_cervezas = no_cervezas – 1

si (dinero != 0)
     imprimir “Vamos por mas cerveza…”
else
     imprimir “Vamonos a dormir….”

FIN
```

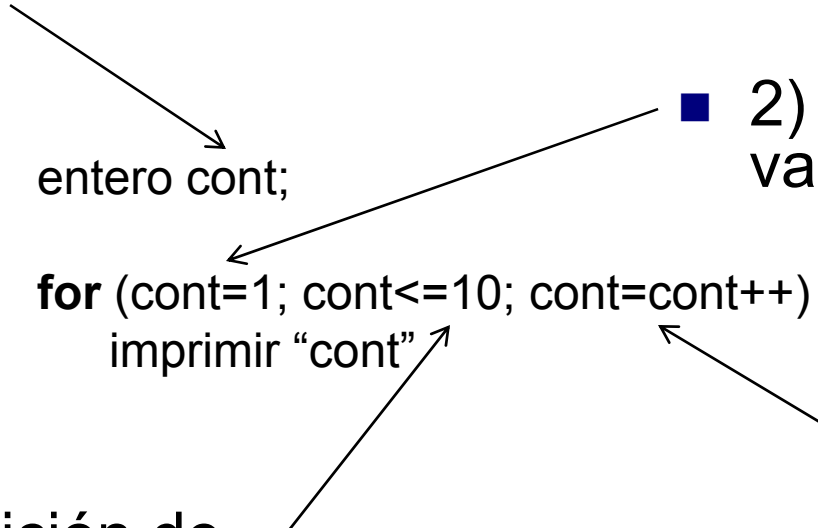


- 1)Variable de control.
- 2) Valor inicial de la variable de control.
- 3) Condición de continuidad.
- 4) Decremento de la variable de control.

Estructura de Repetición for

- Existe en la programación estructurada (lenguaje C) una estructura de repetición que maneja de manera automática los cuatro elementos de la repetición controlada por contador.
- La estructura de repetición: ***for***

- 1)Variable de control.



- 2) Valor inicial de la variable de control.
- 3) Condición de continuidad.
- 4) Decremento de la variable de control.

Equivalencia de for y while

Programa contar hasta 10 con for

```
INICIO

entero cont;

for (cont=1; cont<=10; cont++)
      imprimir “cont”

FIN
```

Programa contar hasta 10 con while

```
INICIO

entero cont;

cont=1;

while (cont<=10)
      imprimir “cont”
      cont++

FIN
```

- Estructuras de repetición controladas por contador (repetición definida).

Ejemplos utilizando la estructura for

```
for (i = 1; i <= 100; i++)

for (i = 100; i >= 1; i--)

for (i = 7; i <= 77; i += 7)

for (i = 2; i <= 20; I += 3)

for (i = 99; i >= 0; I -= 11)
```

Enunciados break y continue

- Existen dos instrucciones que se pueden utilizar para modificar el flujo de control de un programa:
  - ***break***
  - ***continue***

- Cuando se ejecuta la instrucción ***break*** dentro de un ***for, while*** o ***do/while*** este causa la salida inmediata de dichas estructuras. La ejecución del programa continúa con la primera instrucción después de la estructura.

- Cuando se ejecuta la instrucción ***continue*** dentro de un ***while, for*** o ***do/while*** este causa que las instrucciones restantes dentro de la estrucutra sean saltadas y se ejecuta la siguiente iteración del ciclo.

Programa contar hasta

```
INICIO

entero x;
entero i;

imprimir “Dame un número del 0 al 25”
(x<-)

for (i=0; i<=25; i++)
    {
    si (i == x)
            break

    imprimir i
    }

FIN
```

Programa saltarnos un número

```
INICIO

entero x;
entero i;

imprimir “Dame un número del 0 al 25”
(x<-)

for (i=0; i<=25; i++)
    {
    si (i == x)
            continue

    imprimir I
    }

FIN
```

Ejercicio en clase

- Traduzca del seudocódigo al lenguaje C los dos programas anteriores….
- Tiene solo 8 minutos…

Estructuras de Control en C (Prog. Estructurada)

- En total, el lenguaje C tiene **7** estructuras de control:
  - 1 estructura de secuencia.
  - 3 estructuras de decisión (selección).
    - **if**
    - **if/else**
    - **switch**
  - 3 estructuras de repetición.
    - **while**
    - **do/while**
    - **for**
- Estas estructuras de control pueden ser a su vez:
  - Apiladas.
  - Anidadas.

Tarea # 4

- 1) Leer Cápitulo 4 del libro “Como programar en C/C++”.
- 2) Investigue sobre la estructura de decisión SWITCH. Explique formato y funcionamiento.
- 3) Realize los siguientes programas en C:
  - 3.1) Realize un programa que saque el volumén de un cubo, piramide, cono y esfera. El programa debe de funcionar con un while controlado por bandera y la selección con una estructura switch.
  - 3.2) Realize un programa que imprima solo los números primos del 1 al 1,000,000 y diga cuantos hay en total.
  - 3.3) Realize un programa que calcule el valor de ¶ (pi) a partir de la serie infinita: pi=4-4/3+4/5-4/7+4/9…..

    Imprima el valor de pi al ir aumentando el número de terminos de la serie. ¿Cuantos términos se requieren para que pi valga 3.14? ¿3.141? ¿3.1415? ¿3.14159?

Programación Modular

- Con forme el tamaño de un programa aumenta, se vuelve mas complicado su desarrollo y mantenimiento.

- Por tanto, se ha demostrado que la mejor forma para desarrollar y mantener un programa grande es construirlo a partir de piezas menores o módulos. Siendo cada módulo mas fácil de manipular o modificar que el programa completo.

- Los módulos en C se llaman funciones.

- Todo programa en C no es más que un conjunto de funciones, siendo la principal la función ***main().***

Funciones en C

- En el lenguaje C, los programas se escriben combinando nuevas funciones hechas por el programador con funciones ya disponibles en las librerias de C (biblioteca estándar de C) .

- Ejemplos de dichas funciones:
  - ***printf()***
  - ***scanf()***
  - ***pow()***
  - ***sqrt()***

- Nosotros podemos a la vez implementar nuestras propias funciones para tareas específicas. Estas se conocen como: funcines definidas por el programador.

Funciones en C

- Las funciones se invocan mediante una llamada de función.
- La llamada de función especifica el nombre de la misma y proporciona información (en forma de argumentos), que la función llamada necesita a fin de llevar a cabo su tarea designada.
- Las funciones operan de una manera jerárquica donde una función es la función que llama o llamador y la otra es la función llamada.
- Ejem: jefe (main) → empleado (printf)

Funciones Matemáticas de biblioteca en C

- Las funciones matemáticas de biblioteca permiten ejecutar ciertos cálculos matemáticos comunes. (Ver apéndice B).

- Las funciones generalmente son invocadas escribiendo el nombre de la función seguido de la lista de argumentos dentro de paréntisis:

  nombre_función(argumento1, argumento 2, ….)

  sqrt(x) ---- > raiz cuadrada de x

  pow(x, y) ---- > x elevado a la potencia y

- NOTA: todas las funciones de la biblioteca de matemáticas regresan en resultado de tipo ***double***.

Funciones Matemáticas de biblioteca en C

Función	Descripción
sqrt(x)	raíz cuadrada de x
exp(x)	función exponencial e a la x
log(x)	logaritmo natura de x (base e)
log10(x)	logaritmo de x (base 10)
fabs(x)	valor absoluto de x
ceil(x)	redondeo hacia entero mayor que x
floor(x)	redondeo hacia entero menor que x
pow(x,y)	x elevado a la potencia y
fmod(x,y)	residuo de x/y de punto flotante
sin(x)	seno trigonométrico de x (x en radianes)
cos(x)	coseno trigonométrico de x (x en radianes)
tan(x)	tangente trigonométrica de x (x en radianes)
asin(x)	arco cuyo seno es x (resultado en radianes)
acos(x)	arco cuyo coseno es x (resultado en radianes)
atan(x)	arco cuyo tangente es x (resultado en radianes)

- No olvide incluir el archivo de cabecera math.h con la intrucción del preprocesador de C #include<>.

Ejemplo de Funciones Matemáticas en C

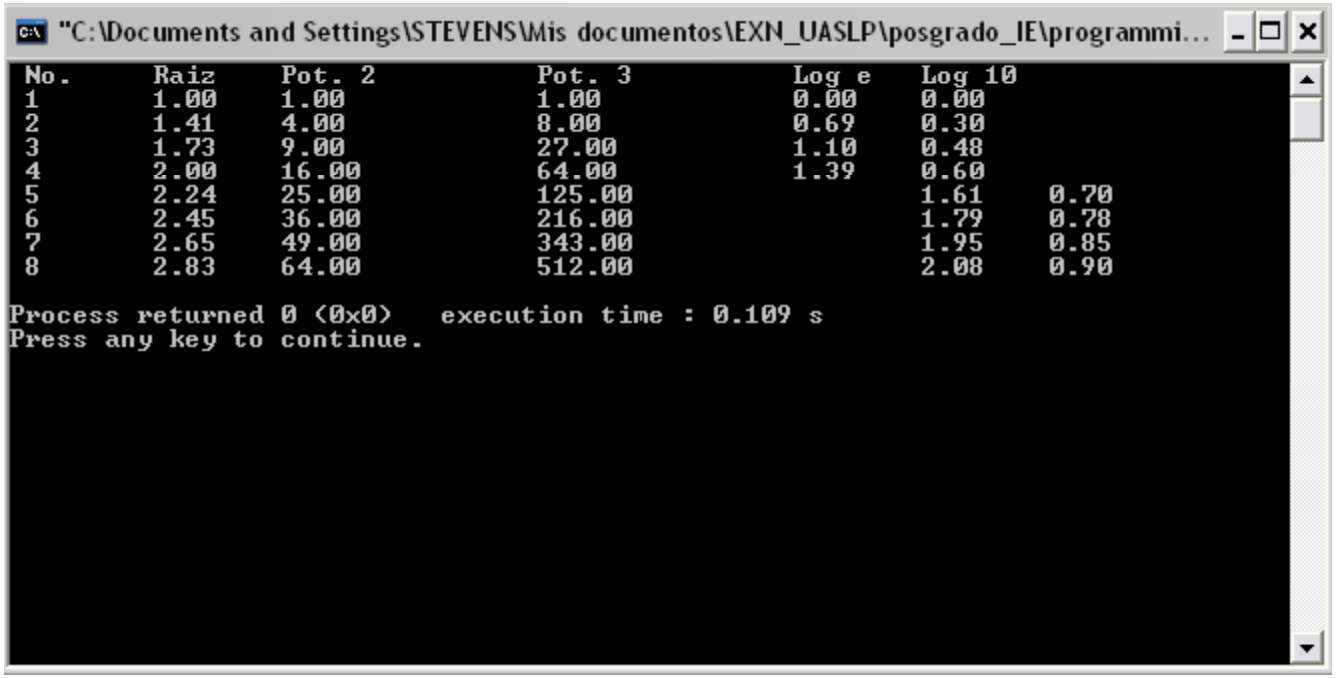
main.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

int main()
{
    float a;
    float b;

    int i;

    // Iniciar variables

    a=0;
    b=0;

    printf(" No. \t Raiz \t Pot. 2 \t Pot. 3 \t Log e \t Log 10 \n");

    for (i=1; i<=8; i++)
    {

        printf(" %d \t %.2f \t %.2f \t\t %.2f \t\t %.2f \t %.2f \n", i, sqrt(i), pow(i,2), pow(i,3), log(i), log10(i));

    }


    return 0;
}
```

Archivo de cabecera math.h

Llamado de funciones con argumentos.

Ejemplo de Funciones Matemáticas en C



- Recuerde que las funciones regresan valores del tipo ***double***.

- ¿Como se modificó el especificador de conversión para solo imprimir dos decimales?

Funciones en C

- En el lenguaje C, en los programas que contengan muchas funciones, se debera programar de manera que la función ***main()*** este organizada con un conjunto de llamadas a funciones que ejecuten la mayor parte del trabajo (calculos) del programa.

- En general, las ventajas de la programación modular en base a funciones son:
  - 1) Se obtiene un código fuente más manipulable y fácil de modificar.
  - 2) La reutilización del software, es decir, el uso de funciones existentes para crear nuevos programas.
  - 3) Evitar en un programa la repetición (duplicación) de un mismo código en un programa.

Funciones en C

- En general, cada función deberá limitarse a ejecutar una tarea sencilla y bien definida y el nombre de la función debera expresar de forma clara dicha tarea.

- Lo anterior, facilita el entendimiento de un programa y promueve la reutilización de código.

```
entero cuadrado( entero x)
{
    regresa (x*x)
}

Entero cubo ( entero x )
{
    regresa (x*x*x)
}
```

- Ejemplo de seudocódigo para funciones.

Funciones en C

Programa cuadrado y cubo con mis super funciones

INICIO

entero x;

imprimir “Dame un número entero:”
(x<-)

Imprimir “El cuadrado del número es:” cuadrado(x)

Imprimir “El cubo del número es:” cubo(x)

FIN

```
entero cuadrado( entero x)
{
     regresa (x*x)
}
```

```
Entero cubo ( entero x )
{
     regresa (x*x*x)
}
```

Llamadas a mis super funciones!

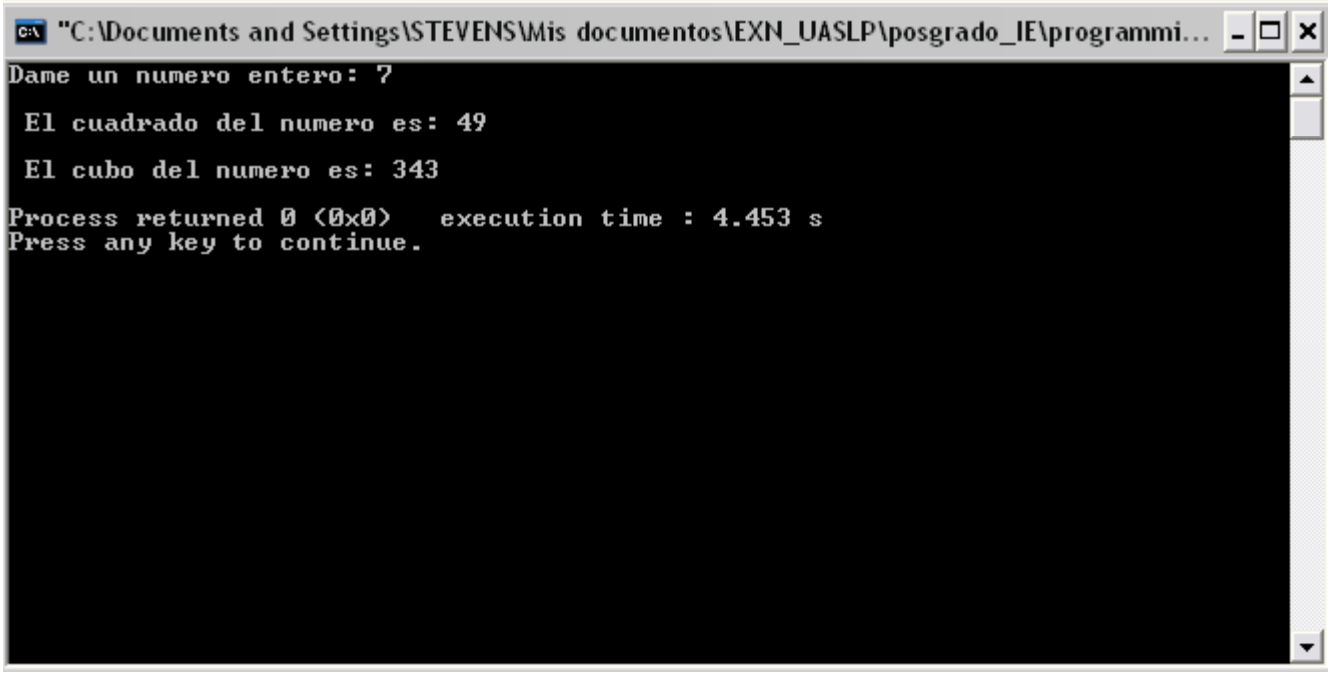
Funciones en C

main.c

```c
#include <stdio.h>
#include <stdlib.h>

// Prototipos de funcion

int cuadrado(int);
int cubo(int);

int main()
{
    int x;

    printf("Dame un numero entero: ");
    scanf("%d", &x);

    printf("\n El cuadrado del numero es: %d \n", cuadrado(x));
    printf("\n El cubo del numero es: %d \n", cubo(x));

    return 0;
}

// Definiciones de las super funciones

int cuadrado(int a)
{
    return (a*a);
    }

int cubo(int a)
{
    return (a*a*a);
    }

```

- Prototipos de funciones
- Llamado de funciones
- Definiciones de funciones

Funciones en C



- ¿y la función main()?
- ¿que es el tipo void?

Prototipos de funciones en C

- Un prototipo de función le indica al compilador:
  - El tipo de dato que regresa la función.
  - El número de parámetros que la función espera recibir.
  - Los tipos de dichos parámetros.
  - El orden en el cual se esperan dichos parámetros.
- El compilador utiliza los prototipos de funciones para verificar que las llamadas a una función sean correctas.
- Los archivos de cabecera (archivos .h) contienen todos los prototipos de funciones de dicha biblioteca.

Temas para Examen #1

- Capítulos 1 al 5 del libro de texto “Como programar en C/C++”.
- En el capítulo 5 hasta antes de números aleatorios.
- Temas importantes:
  - Programación estructurada
  - Seudocódigo
  - Estructuras de control
  - Programación en C

Tipos de llamadas a funciones

- Existen dos formas de invocar (llamar) funciones:
  - Llamada por valor.
  - Llamada por referencia.

- En el caso de la llamada por valor se crea una copia del valor del argumento y ésta se pasa a la función llamada. Las modificaciones al valor de la copia no afectan al valor original de la variable en la función llamadora.

- En el caso de la llamada por referencia, la función llamadora permite que la función llamada modifique el valor de la variable original. Es decir, no se crea ninguna copia del valor del argumento.

Tipos de llamadas a funciones

- La llamada por valor debe ser utilizada siempre que la función llamada no necesite modificar el valor de la variable original en la función llamadora.

- La llamada por referencia debe ser utilizada sólo en funciones llamadas confiables, que necesiten por alguna parte del programa modificar el valor original de la variable en la función llamadora.

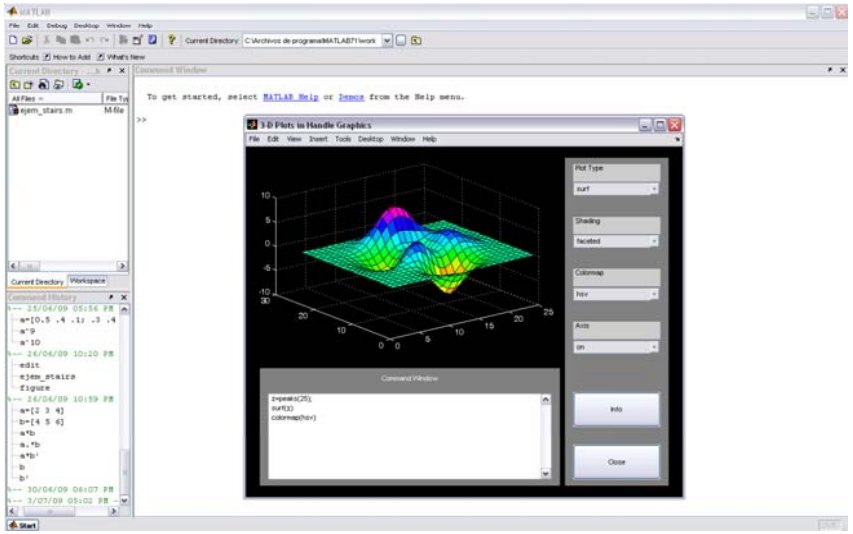
- En general, en el lenguaje C todas las llamadas a funciones son llamadas por valor. Sin embargo, es posible simular llamadas por referencia utilizando operadores de dirección de memoria (& y apuntadores -> Capítulo 7).

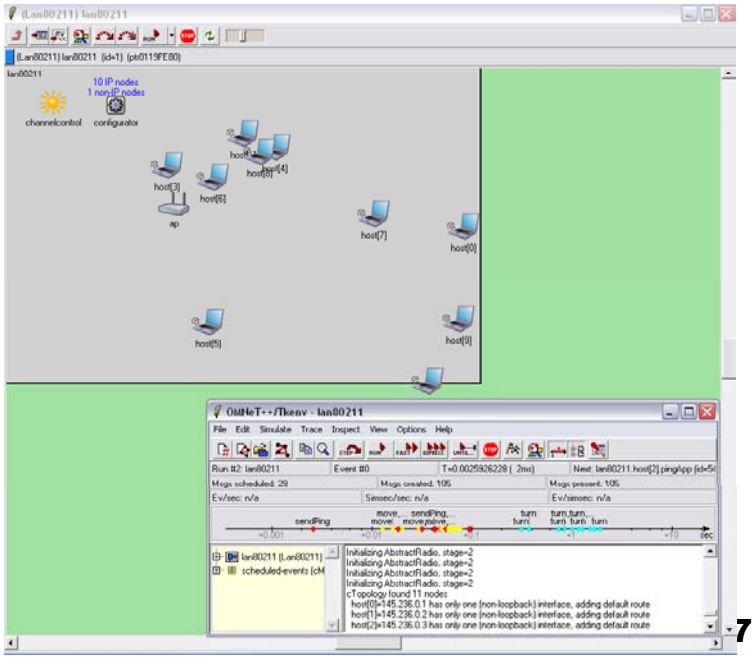
Números Aleatorios en C

- Una de las tareas más util e importante de la programación es la simulación.
- La simulación por computadora es un intento por modelar situaciones de la vida real por medio de un programa.
- Tradicionalmente, el modelado formal de sistemas se realiza mediante un modelo matemático (analítico) para solucionar algún problema de ingeniería aplicada o de investigación científica.
- La simulación por computadora ha sido utilizada para validar modelos matemáticos o incluso para resolver algún problema que no permite una solución analítica.

Números Aleatorios en C

- Algunos ejemplos de simulación por computadora en el area de ingeniería electrónica:
  - Circuitos electricos/electrónicos (analógicos y/o digitales)
  - Sistemas de Control y Automatización
  - Sistemas de Comunicación
  - Redes de Computadoras





Números Aleatorios en C

- Para implementar una simulación por computadora requerimos de números aleatorios.
- La biblioteca estándar de C (stdlib.h) incluye la útil función rand().
- La función rand() genera un entero entre 0 y RAND_MAX (costante definida en stdlib.h). El valor de RAND_MAX debe ser por lo menos 32767. En teoria, al llamar la función rand() obtenemos un entero uniformemente distribuido entre 0 y RAND_MAX.
- Sin embargo, en muchas aplicaciones necesitasmos un rango menor de números aleatorios, por lo tanto, se tiene que adecuar el resultado de la función rand().

Números Aleatorios en C

- Supongamos que estamos en Las Vegas y queremos simular un juego de dados.

- Para simular el dado necesitamos que rand() nos proporcione enteros del 1 al 6 uniformemente distribuidos (con la misma probabilidad de ocurrencia).

```
entero numero,

numero = 1 + (rand() % 6 )
```

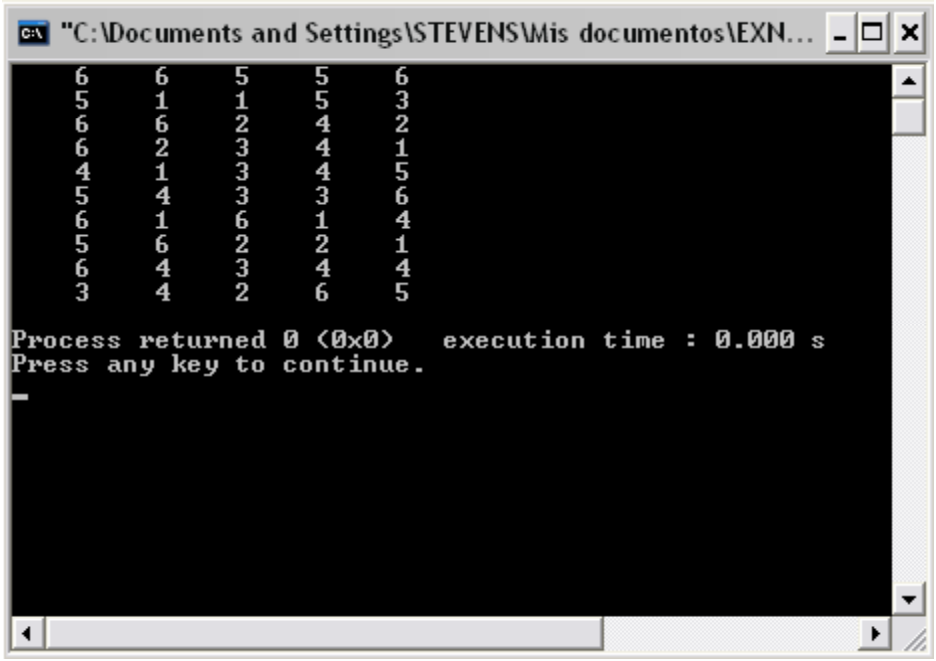
- Factor de desplazamiento
- Factor de dimensionamiento

Números Aleatorios en C

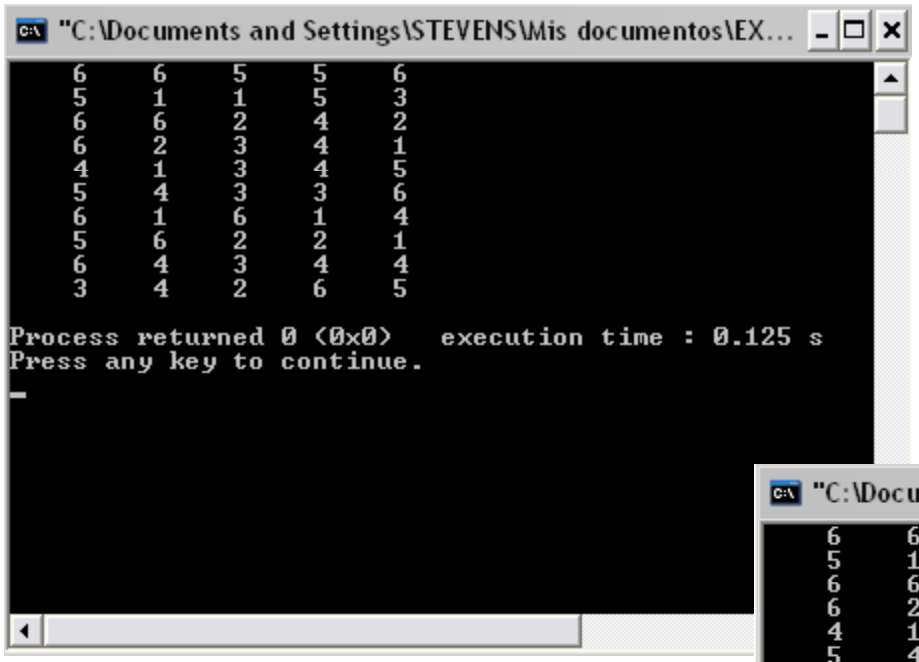
main.c

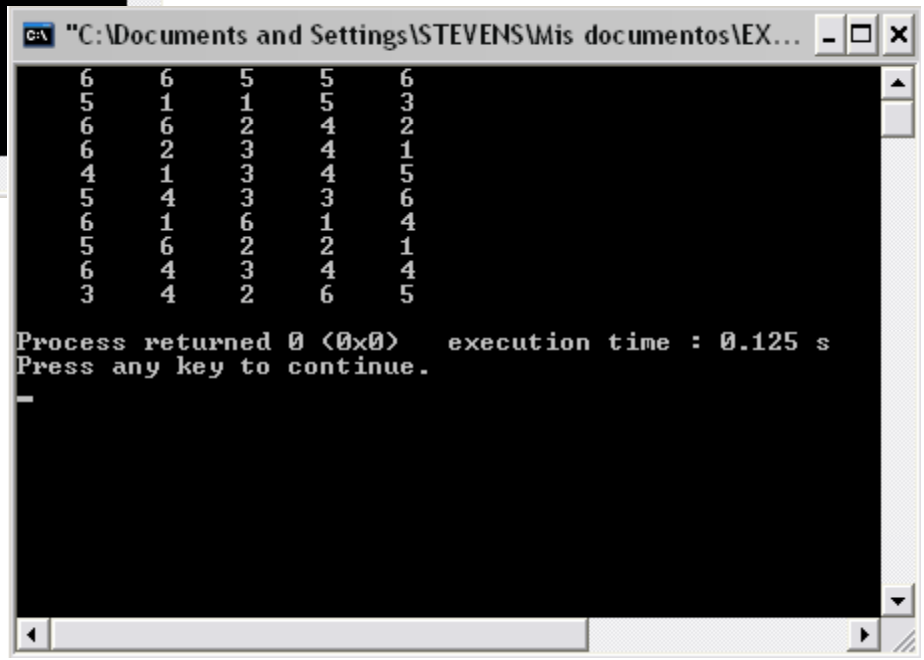
```c
// Programa dado

#include <stdio.h>
#include <stdlib.h>

int main()
{
    int numero;

    for(numero=1; numero <= 50; numero++)
    {
        printf("%5d", 1+(rand() % 6));

        if (numero % 5 == 0)
            printf("\n");
        }


    return 0;
}
```

- ¿Son estos números aleatorios?
- ¿Que pasa si corremos el programa nuevamente?



Números Aleatorios en C





- ¿Son estos números realmente aleatorios?

Números Seudo-Aleatorios en C

- En realidad, la función rand() genera números seudoaleatorios!.
- Cada vez que se llama la función rand(), esta produce una secuencia de números que parecen ser aleatorios. Sin embargo, cada vez que el programa se ejecute la secuencia se repetirá a sí misma.
- Afortunadamente, C permite la generación de números realmente aleatorios. Para tal efecto, se utiliza la función srand() incluida en la misma libreria.
- La función srand() toma un argumento entero del tipo unsigned llamado semilla, para que cada ejecución del programa la función rand() produzca una secuencia de números aleatorios.

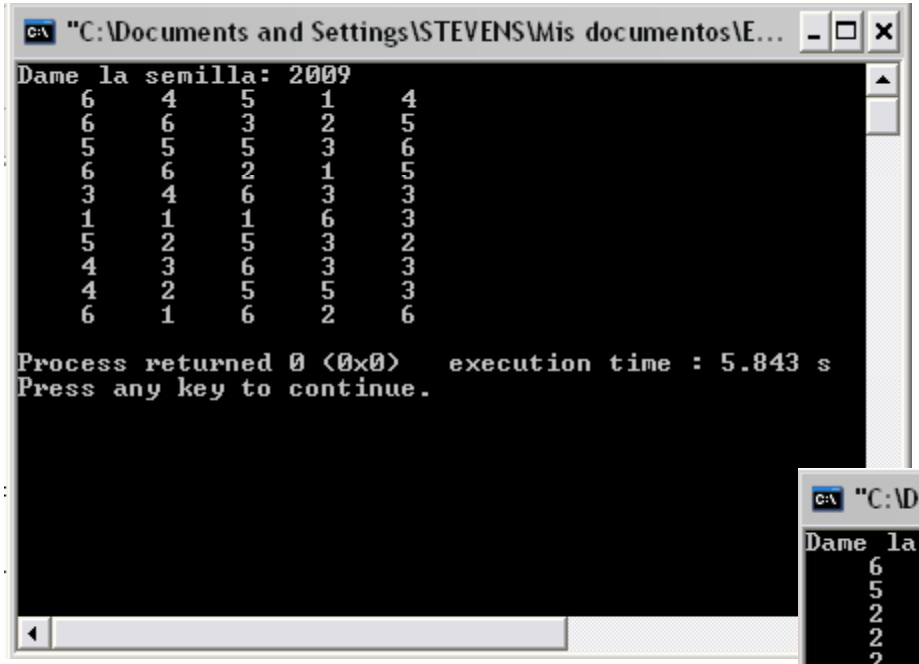
Números Aleatorios en C

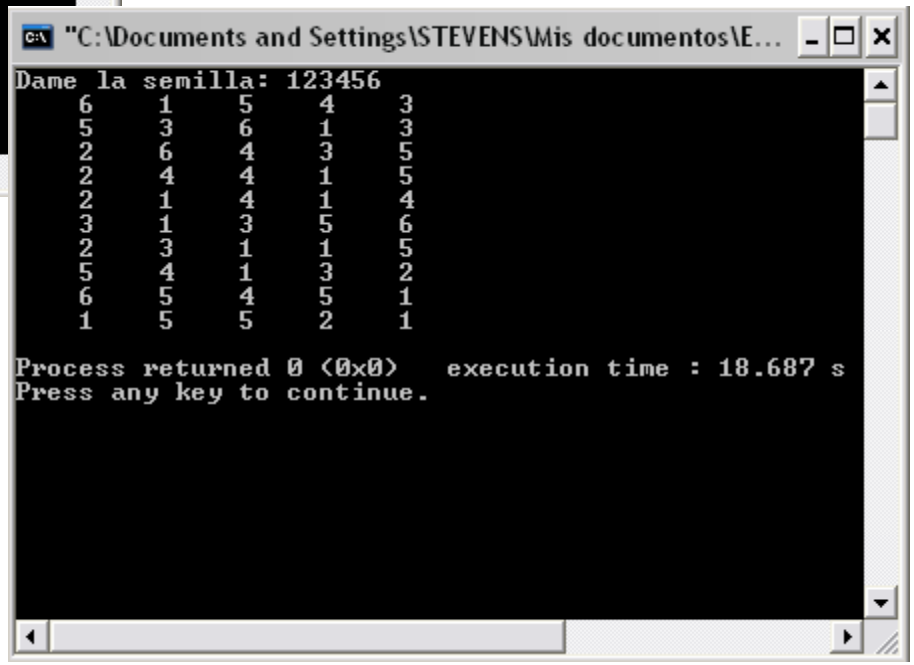
- Las variables del tipo unisgned int de dos bytes pueden tener valores enteros positivos dentro del rango 0 hasta 65535.

main.c

```
1   // Programa dado realmente aleatorio
2
3   #include <stdio.h>
4   #include <stdlib.h>
5
6   int main()
7   {
8       int numero;
9       unsigned semilla;
10
11      printf("Dame la semilla: ");
12      scanf("%u", &semilla);
13      srand(semilla);
14
15      for(numero=1; numero <= 50; numero++)
16      {
17          printf("%5d", 1+(rand() % 6));
18
19          if (numero % 5 == 0)
20              printf("\n");
21          }
22
23
24      return 0;
25  }
26
```

Números Aleatorios en C





Números Aleatorios en C

- Una forma de hacer un programa aleatorio sin tener que introducir cada vez una semilla, es utilizar el enunciado:

```
srand(time(NULL))
```

- Dicha instrucción hace que la computadora lea su reloj interno para obtener de forma automática un valor para la semilla.

- La función time() devuelve la hora actual del día en segundos. El prototipo de función y demas informaicón para usar la función time() se encuentra en la libreria time.h

Resumen de Números Aleatorios en C

- En general, podemos utilizar la siguiente función para generar números aleatorios en C:

```
int numero;

srand(time(NULL));

numero = a + (rand() % b );
```

- Generador automático de semillas.
- Factor de desplazamiento
- Factor de dimensionamiento

Funciones Recursivas

- Hasta el momento, hemos manejado funciones que llaman a otras funciones. Sin embargo, también pueden existir funciones que se llamen asi mismas.

- Este tipo de funciones son las funciones recursivas.

- Una función recursiva es llamada para resolver un problema. La función de hecho solo sabe cómo resolver el caso más simple, es decir el llamado caso base.

- Ejemplos clásicos de funciones recursivas:
  - Factorial de un número.
  - Serie Fibonacci.

Tarea # 5

- 1) Leer Cápitulo 5 del libro “Como programar en C/C++”.
- 2) Realize en C los programas correspondientes a los siguientes problemas del final del cápitulo 5:
  - 5.26
  - 5.31
  - 5.35
  - 5.36
  - 5.37

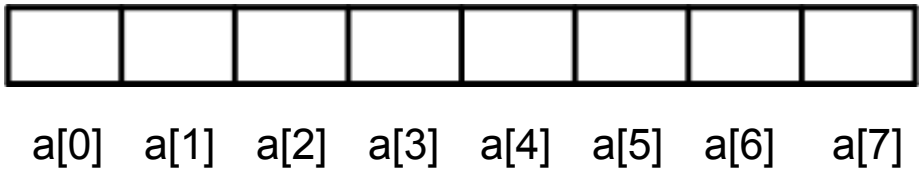
Para entregar: lunes 13 de julio.

Arreglos en C

- Los arreglos son estructuras de datos consistentes en elementos de datos relacionados del mismo tipo.
- Un arreglo es un grupo de posiciones de memoria relacionadas entre sí, por el hecho de que todas tienen el mismo nombre y son del mismo tipo.
- Para hacer referencia a una posición en particular o elemento dentro del arreglo, se especifica el nombre del arreglo y el número de posición del elemento en cuestion dentro del mismo.

Arreglos en C

- Arreglo de enteros (int) llamado a

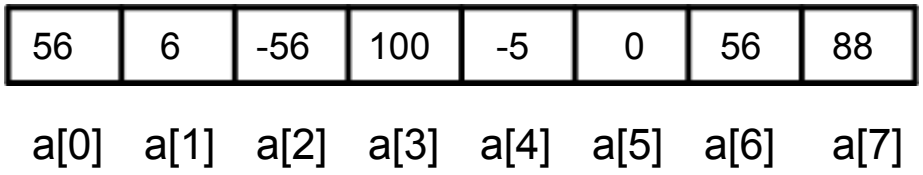


- Este arreglo contiene 8 elementos.
- Cualquier elemento puede ser referenciado individualmente dandole el nombre del arreglo y luego entre [] la posición de dicho elemento.
- El primer elemento de cualquier arreglo es el elemento cero (0).
- NOTA: el primer elemento es 0 tons a[0], el segundo elemento es 1 tons a[1], el tercer elemento es 2 tons a[2], etc.

Arreglos en C

- El número de posición que aparece dentro de [] se le conoce formalmente como subíndice.

- No olvide que el subíndice debe ser siempre un entero o una expresión entera.



- El nombre del arreglo es a.

- ¿cuanto vale a[2]?
- ¿cuanto vale a[2+2]?
- ¿cuanto vale (a[3]-a[0])*a[1]?

Arreglos en C

- Los corchetes [] utilizados para cerrar el subíndice de un arreglo son considerados por el lenguaje C como un operador. Tienen el mismo nivel de precedencia que los paréntesis.

(a[3]+a[2]-a[0])*a[7]

- Los arreglos ocupan espacio en memoria. El espacio depende del tipo del arreglo (int, float, double, etc.) y de la cantidad de elementos.

- Al momento de declarar un arreglo, la computadora reserva la cantidad apropiada de memoria para almacenar el arreglo.

Arreglos en Seudocódigo

```
INICIO
entero cont;

// Declaración del arreglo

entero numeros[100];

// Iniciar el arreglo

for (cont=0; cont<=100; cont++)
      {
      numeros[cont] = 0;
      }

// Imprimir arreglo

for (cont=0; cont<=100; cont++)
      {
      imprimir ( %d \n", numeros[cont] );
      }

FIN
```

- ¿Donde esta el error?

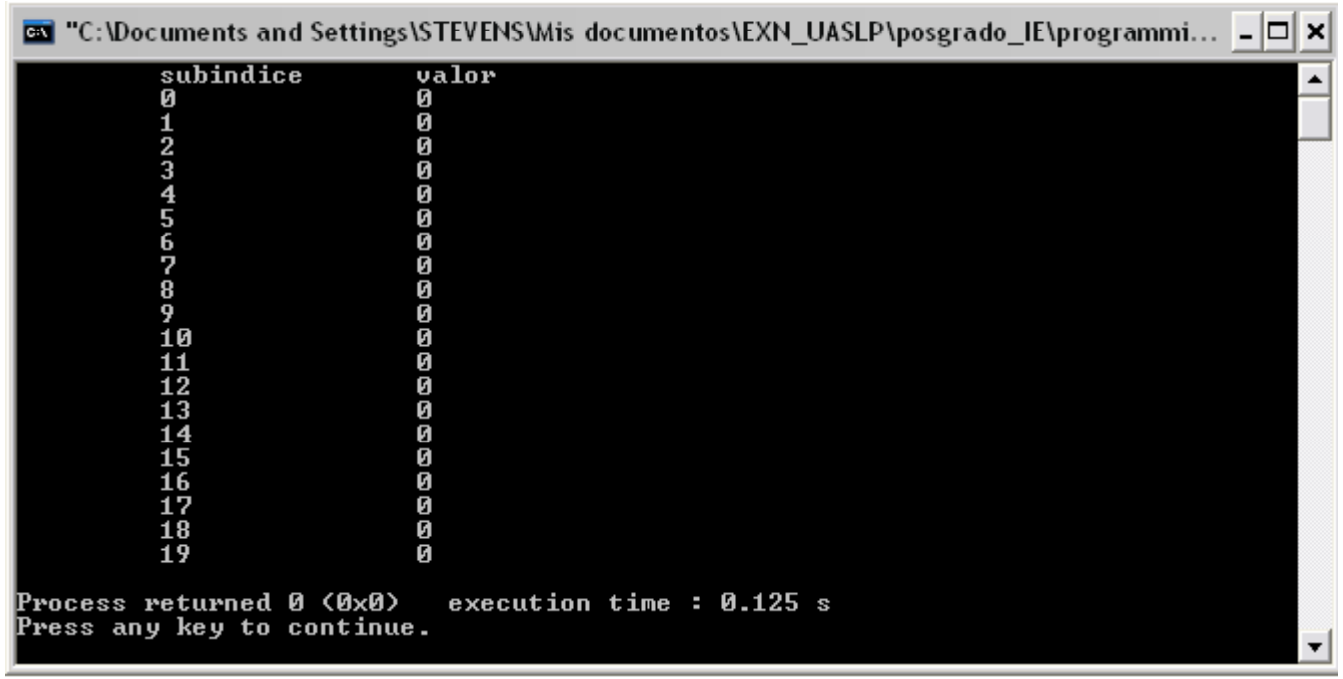
Arreglos en C

main.c

```
1   #include <stdio.h>
2   #include <stdlib.h>
3
4   int main()
5   {
6
7       int cont;
8       int a[20];
9
10      for (cont=0; cont<20; cont++)
11      {
12          a[cont]=0;
13      }
14
15      printf("\t subindice \t valor \n");
16
17      for (cont=0; cont<20; cont++)
18      {
19          printf("\t %d \t\t %d \n", cont, a[cont]);
20      }
21
22      return 0;
23  }
24
```

- Declaración del arreglo.
- Inicialización del arreglo a ceros.
- Impresión del arreglo.

Arreglos en C

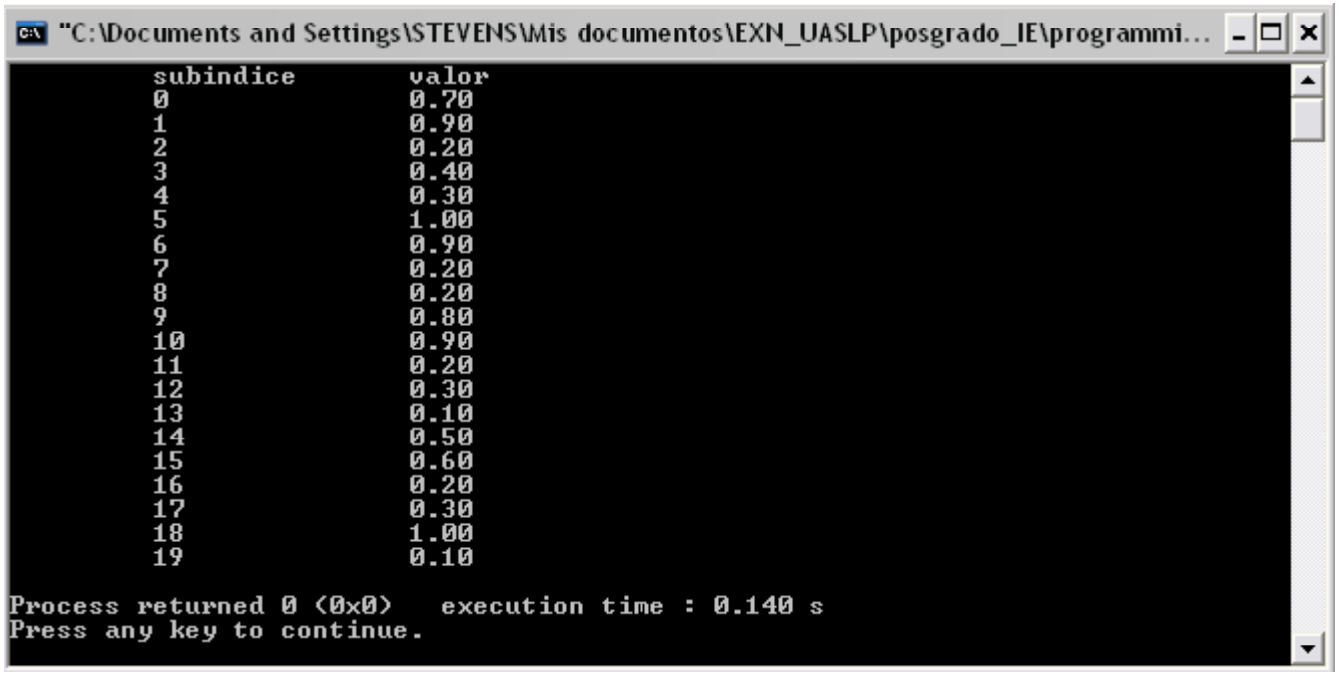


Arreglos en C

main.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int main()
{

    int cont;
    float a[20];
    srand(time(NULL));

    for (cont=0; cont<20; cont++)
    {
        a[cont]=(float)(1+rand() % 10)/10;
    }

    printf("\t subindice \t valor \n");

    for (cont=0; cont<20; cont++)
    {
        printf("\t %d \t\t %.2f \n", cont, a[cont]);
    }

    return 0;
}
```

Arreglos en C



Arreglos en C

- Los elementos de un arreglo también pueden ser inicializados en la declaración del arreglo mismo, declaración del arreglo seguido de una lista de valores separados por comas llamados inicializadores.

- NOTA: Si dentro del arreglo existe un número menor de inicializadores que de elementos, los elementos restantes son inicializados a cero de forma automática.

```
int numeros[5] = {1, 2, 3, 4, 5};

int numeros[5] = {1, 2, 3};

int numeros[5] = {0};

int numeros[5] = {1};

int numeros[5];
```

Ejercicio en clase

- Tiene los siguientes arreglos:
  - int a[4] = {0};
  - int b[3] = {5};
  - int c[5] = {1,2,3,4,5};
  - (a[0]-c[0])*b[2] =
  - (c[0]+c[2])*b[0] =
  - (a[3]+a[2]+c[a[3]])*(c[4]-c[a[2]]+b[2]) =
  - (b[a[3]+b[2]]+c[c[0]+c[1]])*c[b[0]-c[1]] =

Arreglos en C

- Los arreglos en C pueden tener múltiples subíndices (es decir, múltiples dimensiones).
- Una de la utilización más común de los arreglos con múltiples subíndices es para representar matrices, consistiendo la información arreglada en renglones y columnas.
- Para identificar un elemento particular dentro de la matriz se deben especificar dos subíndices, el primero en general representa al renglón del elemento y el segundo identifica a la columna del elemento.
- Se les conoce también como arreglos de doble subíndice.

Arreglos en C

- Una versión de C estándar debe soportar por lo menos arreglos de 12 subíndices.

- En el caso de un arreglo de doble subíndice de 3X3:

		columna 0	columna 1	columna 2
	0	(0,0)	(0,1)	(0,2)
renglón	1	(1,0)	(1,1)	(1,2)
	2	(2,0)	(2,1)	(2,2)

Arreglos en C

- Para hacer referencia a los elementos del arreglo:

```
int a[3][3] = {{1,2,3},{4,5,6},{7,8,9}};
```

columna

renglón	0	1	2
0	a[0][0]	a[0][1]	a[0][2]
1	a[1][0]	a[1][1]	a[1][2]
2	a[2][0]	a[2][1]	a[2][2]

- a[0][0]+a[2][2] =
- a[2][2]+a[0][2]=

Repasando Algebra Matricial en C

- La suma de dos matrices cuadradas A + B:
  - Matriz A

1	2	3
4	5	6
7	8	9

  - Matriz B

1	0	0
0	1	0
0	0	1

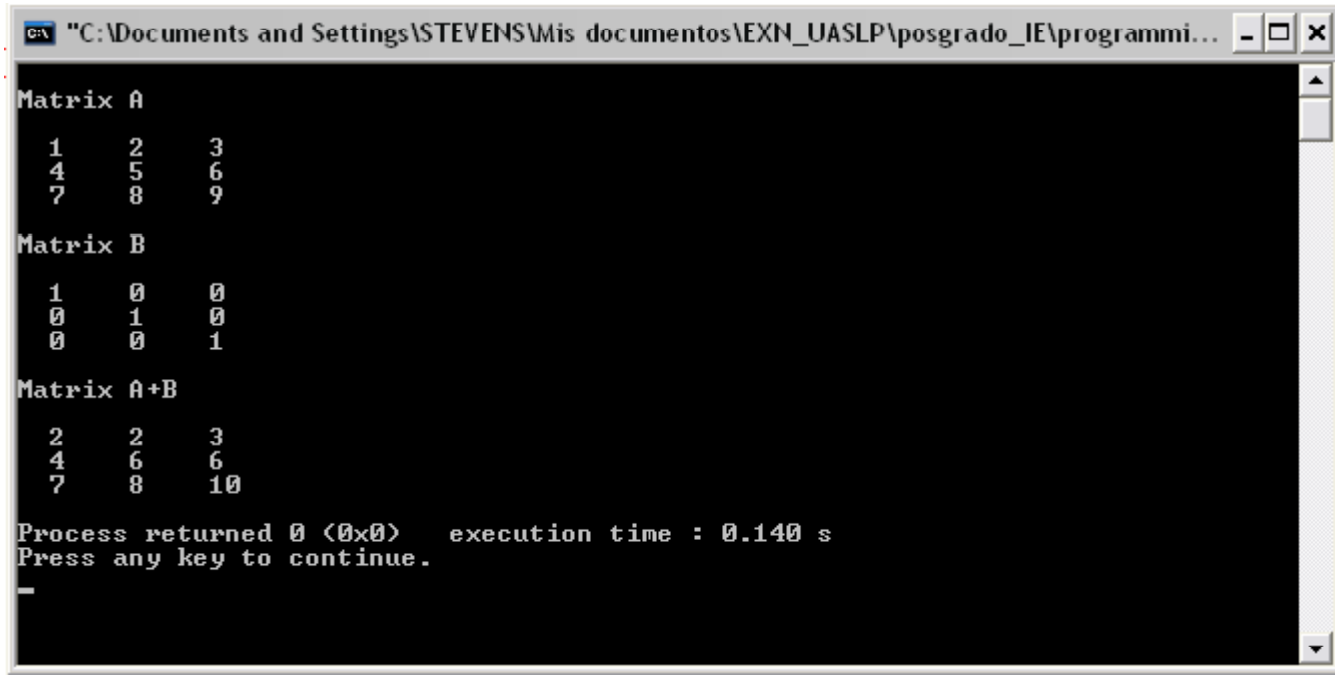
- Resolvamos el problema en C….

Repasando Algebra Matricial en C

main.c

```c
#include <stdio.h>
#include <stdlib.h>

int main()
{
    int a[3][3] = {{1,2,3},{4,5,6},{7,8,9}};
    int b[3][3] = {{1,0,0},{0,1,0},{0,0,1}};
    int i,j;

    printf("\n");
    printf("Matrix A\n\n");

    for(i=0; i<3; i++)
    {
        for(j=0; j<3; j++)
        {
            printf("  %d  ", a[i][j]);
        }
        printf("\n");

    }

    printf("\n");
    printf("Matrix B\n\n");

    for(i=0; i<3; i++)
    {
        for(j=0; j<3; j++)
        {
            printf("  %d  ", b[i][j]);
        }
        printf("\n");

    }

    printf("\n");
    printf("Matrix A+B\n\n");

    for(i=0; i<3; i++)
    {
        for(j=0; j<3; j++)
        {
            printf("  %d  ", (a[i][j]+b[i][j]));
        }
        printf("\n");

    }
    return 0;
}
```

Repasando Algebra Matricial en C



Tarea # 6

- 1) Leer Cápitulo 6 del libro “Como programar en C/C++”.
- 2) Realize en C los programas correspondientes a los siguientes problemas de la tarea #1 del curso propedéutico de algebra matricial:
  - 1.a (productos)
  - 1.b (productos)
  - 4.a (determinante)
  - 5.a (inversa)

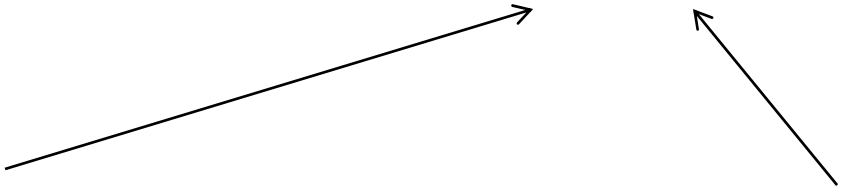
  - Es recomendable que su implementación del programa funcione para cualquier tamaño de matriz. Asi podrá utilizar su código en futuros programas (tareas).

Para entregar: lunes 13 de julio.

Arreglos como parametros de funciones

- Para pasar un arreglo como argumento de una función, se especifica unicamente el nombre del arreglo.

```
int numeros[20] = {0};

int contar_numeros(numeros, 20)
```

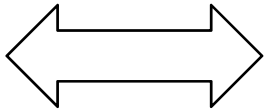


- Nombre del arreglo.
- Tamaño del arreglo.

Arreglos como parametros de funciones

- El lenguaje C pasa de forma automática los arreglos a las funciones utilizando llamadas por referencia. Por lo tanto, las funcines llamadas puden modificar los valores originales de los elementos del arreglo.

- La razón es que el nombre del arreglo es de hecho la dirección en memoria del primer elemento de dicho arreglo:

numeros  &numeros[0]

Arreglos como parametros de funciones

- Para que una función reciba un arreglo a través de una llamada de función, la lista de parámetros debe indicar que se va a recibir un arreglo. La definición de nuestra función **numeros()** seria:

```
int contar_numeros(int nums[], int tamaño)
{
      int i;
      int suma=0;

      for(i=0; i<tamaño; i++)
            suma=suma+nums[i];
      return suma;
}
```

- Y el prototipo de función:

```
int contar_numeros(int [], int)
```

Arreglos como parametros de funciones

- Note que si solo se pasa un elemento del arreglo este pasa por llamada por valor, cuando se pasa todo el arreglo este pasa por llamada por referencia.

- Si se quiere asegurar que la función llamada no modifique los valores originales del arreglo se puede utilizar el calificador especial **const** tanto en la definición y el prototipo de la función.

- Los valores quedan protegidos a cambios, cualquier intento para modificar un elemento del arreglo dentro de la función da como resultado en error al momento de compilar el programa.

Arreglos como parametros de funciones

- Para pasar arreglos de doble subíndice a una función notese que el primer subíndice no se require pero todos los demas subíndices SI son requeridos en su definición y prototipo.

```
int contar_numeros2D(int [][3])
```

- Prototipo de función.

```
int numeros2D[3][3]={0};
```

- Declaración de arreglo 2D.

```
int contar_numeros2D(int nums2D[][3])
```

- Definición de función.

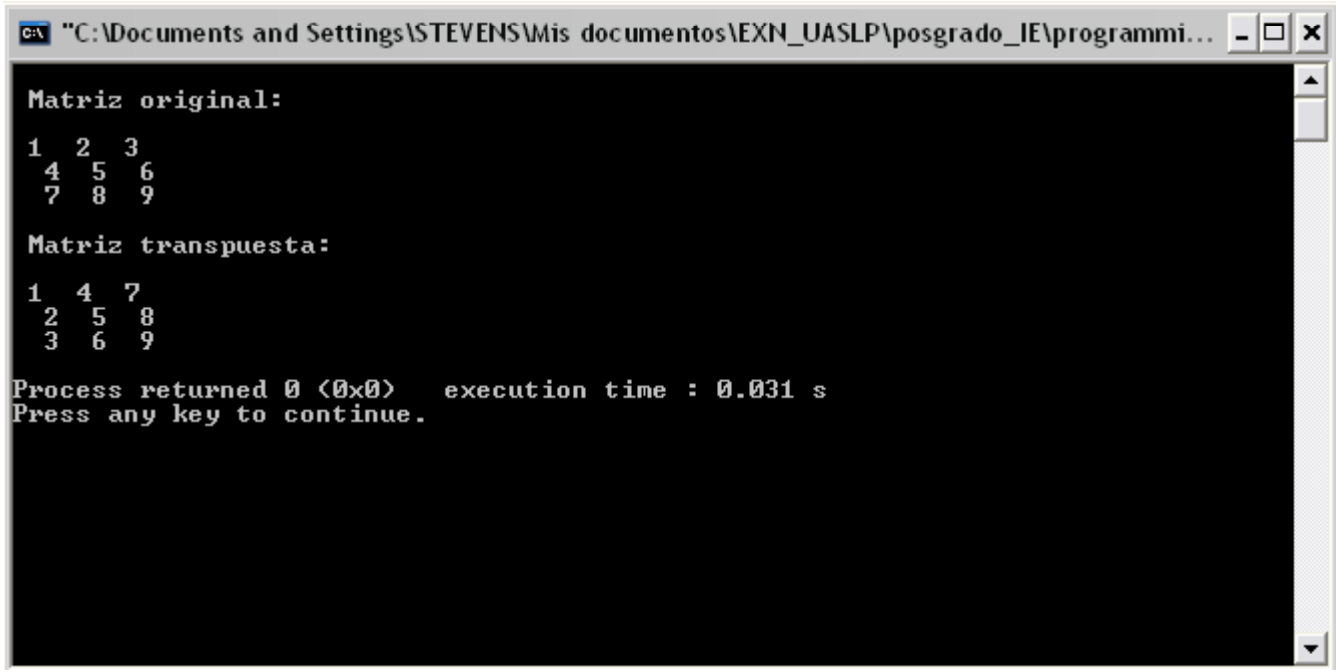
- ¿porque no require el primer subíndice?
- ¿porque si require los demas subíndices?

main.c

```c
#include <stdio.h>
#include <stdlib.h>

void imprimir_matrix(int[][3], int, int);
void transpose(int[][3], int, int);

int main()
{

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

    printf("\n Matriz original: \n\n");
    imprimir_matrix(matrixA,3,3);

    printf("\n Matriz transpuesta: \n\n");
    transpose(matrixA,3,3);

    return 0;
}

void imprimir_matrix(int a[][3], int ren, int col)
{
    int i,j;

    for (i=0; i<ren; i++)
    {
        for (j=0; j<col; j++)
        {
            printf(" %d ", a[i][j]);
        }
        printf("\n ");
    }
}

void transpose(int a[][3], int ren, int col)
{
    int i,j;
    int b[ren][col];

    for (i=0; i<ren; i++)
    {
        for (j=0; j<col; j++)
        {
            b[i][j]=a[j][i];
        }
    }
    imprimir_matrix(b,ren,col);
}
```

- Prototipos de función.
- Declaración de arreglo 2D.
- Llamadas a funciónes.
- Definiciónes de funciónes.

Arreglos como parametros de funciones



Temas para Examen #2

- Capítulos 1 al 6 del libro de texto “Como programar en C/C++”.

- Temas importantes:
  - Programación estructurada
  - Estructuras de control (IF, IF/ELSE, SWITCH, WHILE, DO/WHILE, FOR)
  - Funciones
  - Generación de números aleatorios.
  - Arreglos
  - Programación en C

Formato Examen #2

- 1a Parte: Examen escrito de la teoria en el salón el miercoles 15 de julio.
- 2a Parte: Proyecto final de programación para entregar el viernes 17 de julio.
- Las dos partes forman la calificación total del examen #2.

2a Parte del Examen #2

- Experimentos para medición de desempeño computacional.
- OBJETIVO
  - Realizar mediciones de tiempo de computo en diferentes procesadores utilizando la inversión de matrices aleatorias.
- PROCEDIMIENTO
  - Usando su programa de inversión de matrices de la tarea #6 encuentre la matriz inversa de la mayor cantidad y tamaño de matrices generadas aleatoriamente.

2a Parte del Examen #2

- PROCEDIMIENTO DETALLADO:
  - 1) Genere una matriz de numeros aleatorios.
  - 2) Verifique si la matriz es invertible.
  - 3) Si lo es, obtenga la inversa.
  - 4) Si no lo es, genere una nueva matriz.
  - 5) Repita el procedimiento la mayor cantidad de veces posible y para diferentes tamaños de matrices. La cantidad y tamaño sera variable según la implementación de cada programa.
  - 6) Repita todo el procedimiento nuevamente en cuando menos 3 procesadores diferentes.
  - 7) Realice un reporte tecnico detallado lo más parecido a un articulo científico. Incluya detalles, decisiones y presente adecuadamente los resultados de preferencia en figuras o tablas.

2a Parte del Examen #2

- La 2a parte del examen se puede hacer en equipo de máximo 2 personas.
- El formato del reporte se especifica a detalle y será proporcionado por el profesor.
- Incluya al final una sección APENDICE con todo su código fuente.
- Se evaluara la calidad del trabajo, claridad y presentación, merito tecnico, relevancia de los resultados y conclusiones.