Cap. 3 - Desarrollo de Programa Estructurado

Esquema

3.1 Introducción

- Antes de escribir un programa:
  - Tener una comprensión profunda del problema
  - Un enfoque cuidadosamente planeado para resolverlo

- Mientras escribiendo un programa:
  - Saber qué "bloques de construcción" están disponibles
  - Usar buenos principios de programación

3.2 Algoritmos

- Problemas de computación
  - Todo se puede resolver ejecutando una serie de acciones en un orden específico
- Algoritmo: procedimiento en términos de
  - *Acciones* a ser ejecutadas
  - *El orden* en que estas acciones deben ser ejecutadas
- Control de programa
  - Especificar el orden de ejecución de las declaraciones

3.3 Pseudocódigo

- Pseudocódigo
  - Lenguaje artificial e informal que nos ayuda a desarrollar algoritmos
  - Similar al idioma cotidiano
  - No se ejecutan realmente en las computadoras
  - Nos ayuda a "pensar" un programa antes de escribirlo
    - Fácil de convertir en un programa C correspondiente
    - Consiste sólo en declaraciones ejecutables

3.4 Estructuras de Control

- Ejecución secuencial
  - Declaraciones ejecutadas una tras otra en el orden escrito
- Transferencia de control
  - Cuando la siguiente declaración ejecutada no es la siguiente en la secuencia
  - El uso excesivo de *goto* condujo a muchos problemas
- Bohm y Jacopini
  - Todos los programas escritos en términos de 3 estructuras de control
    - Estructura de la secuencia: Construida en C. Programas ejecutados secuencialmente por defecto.
    - Estructuras de selección: C tiene tres tipos- **`if`**, **`if/else`**, y **`switch`** .
    - Estructuras de repetición: C tiene tres tipos - **`while`**, **`do/while`**, y **`for`**.
    - Hay palabras claves en C

3.4 Estructuras de Control (II)

- Diagrama de flujo (Flowchart)
  - Representación gráfica de un algoritmo
  - Dibujado usando ciertos símbolos de propósito especial conectados por flechas llamadas *líneas de flujo*.
  - Símbolo del rectángulo (símbolo de acción): indica cualquier tipo de acción.
  - Símbolo ovalado: indica el comienzo o el final de un programa, o una sección de código (círculos).
- Estructuras de control de una sola entrada/salida
  - Conectar el punto de salida de una estructura de control con el punto de entrada de la siguiente (apilamiento de la estructura de control).
  - Hace que los programas sean fáciles de construir.

3.5 La Estructura de Selección Si (`if`)

- Estructura de Selección:
  - Se utiliza para elegir entre cursos de acción alternativos
  - Pseudocodigo:

    Si *la calificación del estudiante es mayor o igual a 60* **entonces**
    Imprinit "Aprobado"
    Fin Si

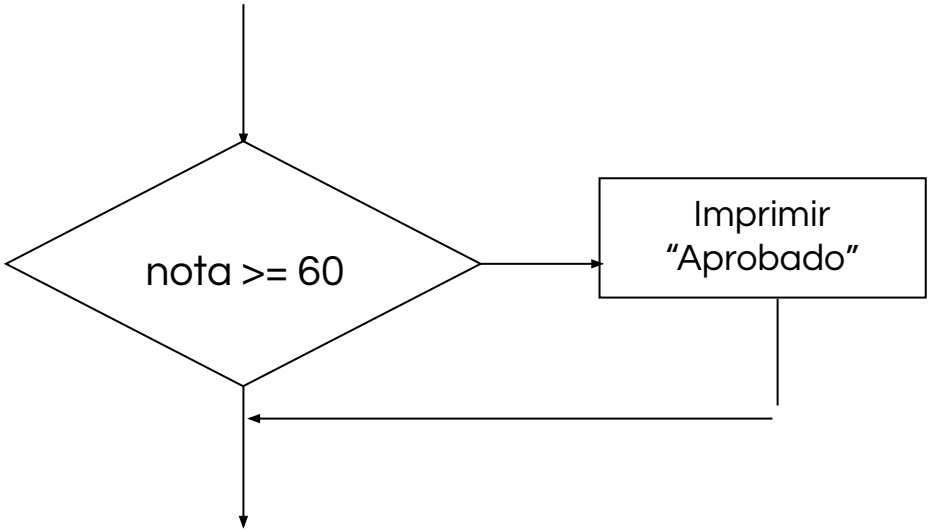
- Condición Verdadera de Si (If condition **`true`**)
  - Imprime la declaración ejecutada y el programa pasa a la siguiente declaración.
  - Si **`falso`**, se ignora la declaración impresa y el programa pasa a la siguiente declaración.
  - La sangría hace que los programas sean más fáciles de leer
    - C ignora los caracteres de los espacios en blanco.
- Declaración de pseudocódigo en C:

```
if ( nota >= 60 ){
   printf( "Aprobado \n" );
}
```

  - El código C se corresponde estrechamente con el pseudocódigo

3.5 La Estructura de Selección Si (`if`) (II)

- Símbolo del diamante (símbolo de decisión) - indica que se debe tomar una decisión
  - Contiene una expresión que puede ser verdadera o falsa (**`true`** or **`false`**)
  - Pruebe la condición, siga el camino apropiado
- La estructura `Si` (**`if`**) es un estructura de una sola entrada/salida.



Se puede tomar una decisión sobre cualquier expresión.

cero - **`falso`**

distinto a cero - **`verdadero`**

Example:

`3 - 4` es **`verdadero !!!`**

3.6 La estructura de selección SI/SINO (`if/else`)

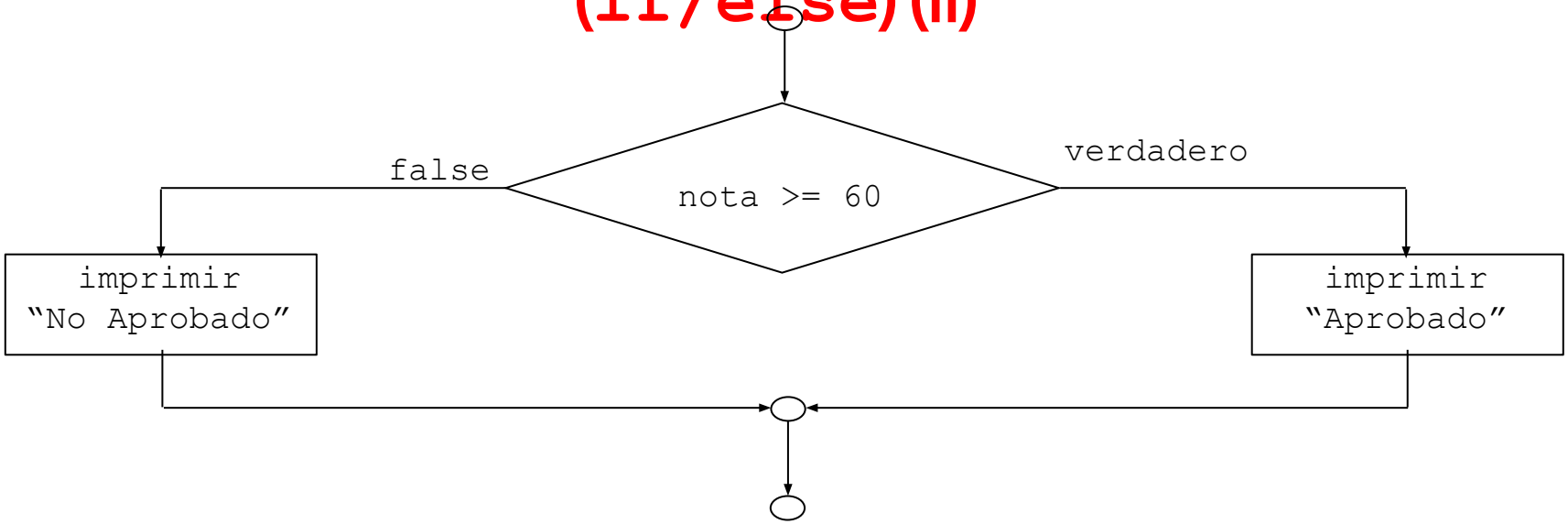
- **`Si (if)`**
  - Solo realiza la acción si la condición es verdadera (**`true`**).
- **`Si/Sino (if/else)`**
  - Una acción diferente cuando la condición es verdadera que cuando es falsa.
- Pseudocodigo:

```
Si la calificación del estudiante es mayor o igual a 60 entonces
    Imprimir "Aprobado"
Sino
    Imprimir "No aprobado"
Fin Si
```

  - Note las convenciones de espaciamiento/indentación
- Código C:

```
if ( nota >= 60 ){
        printf( "Aprobado \n");
}else{
        printf( "No aprobado \n");
}
```


3.6 La estructura de selección SI/SINO (`if/else`) (II)



- Operador Condicional Ternario (`?:`)
  - Toma tres argumentos (condition, valor Si **`true`**, valor Si **`false`**)
  - Nuestro pseudocódigo puede ser escrito:

```
printf( "%s\n", nota >= 60 ? "Aprobado" : "No Aprobado" );
```

o

```
nota >= 60 ? printf("Aprobado\n") : printf("No aprobado\n");
```


3.6 La estructura de selección SI/SINO (`if/else`) (III)

- Estructuras **`Si/Sino`** (**`if/else`**) anidadas
  - Prueba para múltiples casos colocando las estructuras de selección **`if`**/**`else`** dentro de las estructuras de selección **`if`**/**`else`**.

```
Si calificación es mayor o igual a 90 entonces
  Imprimir "A"
Sino Si calificación es mayor o igual a 80 entonces
       Imprimir  "B"
  Sino Si calificación es mayor o igual a 70 entonces
            Imprimir "C"
       Sino Si calificación es mayor o igual a 60 entonces
                 Imprimir "D"
            Sino
                 Imprimir  "F"
```

  - Una vez que se cumple la condición, el resto de las declaraciones se saltan
  - La indentación profunda no suele utilizarse en la práctica

3.6 La estructura de selección SI/SINO (`if/else`) (IV)

- Declaración compuesta:
  - Conjunto de declaraciones dentro de un par de llaves
  - Ejemplo:

```
if ( calificación >= 60 ){
   printf( "Aprobado.\n" );
} else {
   printf( "No Aprobado.\n" );
   printf( "Debes tomar este curso de nuevo.\n" );
}
```

  - Sin las llaves,

```
printf( "Debes tomar este curso de nuevo.\n" );
```

    se ejecutaría automáticamente

- Bloque: declaraciones compuestas con declaraciones

3.6 La estructura de selección SÍ/SINO (`if/else`) (V)

- Errores de sintaxis
  - Detectado por el compilador

- Errores de lógica:
  - Tienen su efecto en el momento de la ejecución
  - No fatal: el programa se ejecuta, pero tiene una salida incorrecta
  - Fatal: el programa sale prematuramente

3.7 La Estructura de Repetición `mientras` (`while`)

- Estructura de Repetición
  - El programador especifica una acción a ser repetida mientras alguna condición permanece **`verdadera`**
  - Pseudocódigo:

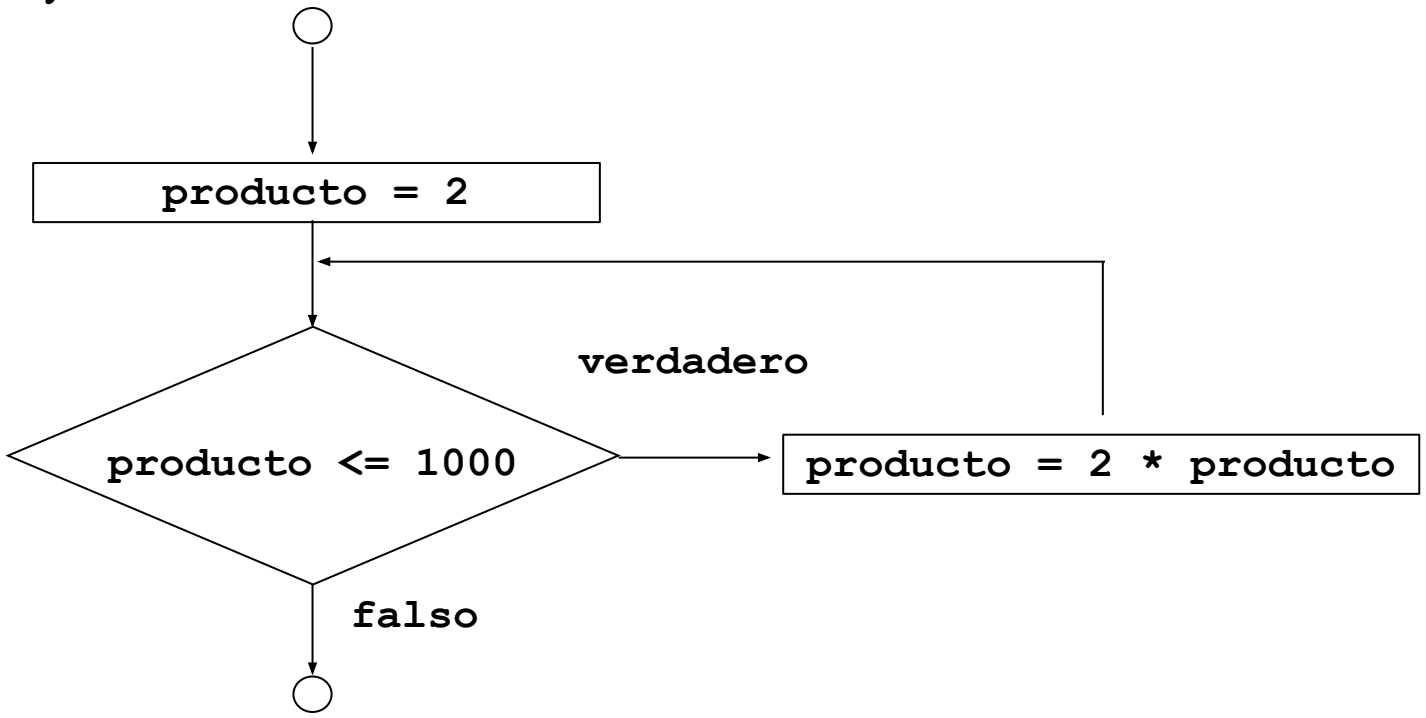
```
Mientras hay más items en mi lista de compras hacer
    Comprar siguiente ítem y tacharlo de mi lista
Fin Mientras
```

  - **`mientras`** que el bucle se repite hasta que la condición se vuelve **`falsa`**

3.7 La Estructura de Repetición `mientras` (`while`) (II)

- Ejemplo:

```
int producto = 2;
while ( producto <= 1000 ){
   producto = 2 * producto;
}
```



3.8 Formulación de Algoritmos (Repetición controlada por contador)

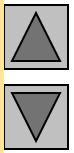
- Repetición controlada por Contador
  - El bucle se repite hasta que el contador alcanza un cierto valor.
  - Repetición definida: se conoce el número de repeticiones
  - Ejemplo: Una clase de diez estudiantes hizo un examen. Las calificaciones (números enteros en el rango de 0 a 100) para este examen están disponibles para usted. Determina el promedio de la clase en el examen.
- Pseudocodigo:

Fijar total a cero
Fijar el contador de calificación a uno

`Mientras` *contador de calificación es menor o igual a diez* **`hacer`**
    Ingrese la siguiente calificación
    Sumar la calificación con el total
    Sumar uno al contador de calificación

`Fin Mientras`

Determinar el promedio de la clase dividiendo total por diez
Imprimir el promedio de la clase



Outline

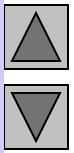
```c
/* Fig. 3.6: fig03_06.c
   programa promedio de la clase con
   repetición controlado por contador */
#include <stdio.h>

int main()
{
   int contador, calificacion, total, promedio;

   /* fase de inicialización */
   total = 0;
   contador = 1;

   /* fase de procesamiento */
   while ( contador <= 10 ) {
      printf( "Ingrese calificación: " );
      scanf( "%d", &calificacion );
      total = total + calificacion;
      contador = contador + 1;
   }

   /* fase de terminación */
   promedio = total / 10;
   printf( "El promedio de la clase es %d\n", promedio);

   return 0; /*Indica que el programa ha terminado con éxito*/
}
```

1. Variables inicializadas

2. Ejecutar el bucle

3. Resultados de salida

```
Ingrese calificación: 98
Ingrese calificación: 76
Ingrese calificación: 71
Ingrese calificación: 87
Ingrese calificación: 83
Ingrese calificación: 90
Ingrese calificación: 57
Ingrese calificación: 79
Ingrese calificación: 82
Ingrese calificación: 94
El promedio de la clase es 81
```



Outline

Salida del programa

3.9 Formulación de algoritmos con refinamiento de arriba hacia abajo, paso a paso (repetición controlada por centinelas)

- El problema se convierte:

  Desarrollar un programa de promedio de clase que procese un número arbitrario de notas cada vez que se ejecute el programa..

  – Número desconocido de estudiantes
  – ¿Cómo sabrá el programa cómo terminar?

- Usar el valor centinela
  – También llamado valor de la señal, valor ficticio o valor bandera
  – Indica "fin de la entrada de datos".
  – El bucle termina cuando se introduce el centinela
  – Valor del centinela es elegido de manera que no pueda ser confundido con una entrada regular (como -1 en este caso)

3.9 Formulación de algoritmos con refinamiento de arriba hacia abajo, paso a paso (repetición controlada por centinelas)(II)

- Refinamiento de arriba a abajo, paso a paso
  - Comienza con una representación en pseudocódigo de la parte *superior*:

    Determinar el promedio de la clase para la prueba
  - Dividir la parte superior en tareas más pequeñas y enumerarlas en orden:

    Iniciar las variables

    Introduzca, sume y cuente las notas de los exámenes

    Calcula e imprime el promedio de la clase
- Muchos programas tienen tres fases
  - Inicialización: inicializa las variables del programa
  - Procesamiento: introduce los valores de los datos y ajusta las variables del programa en consecuencia
  - Terminación: calcula e imprime los resultados finales
  - Esto ayuda a la ruptura de los programas para el refinamiento de arriba hacia abajo

3.9 Formulación de algoritmos con refinamiento de arriba hacia abajo, paso a paso (repetición controlada por centinelas) (III)

- Refinar la fase de inicialización de *Inicializar variables* a

  Iniciar el total a cero
  Iniciar el contador a cero

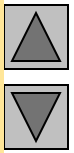
- Refine *la entrada, sume y cuente las notas de los exámenes* para

  Introduce la primer nota (posiblemente el centinela)
  Mientras *que el usuario aún no haya ingresado al centinela* **hacer**
  Sume esta calificación al total
  Añade uno al contador de notas
  Introduce la siguiente nota (posiblemente el centinela)
  Fin Mientras

- Refinar *Calcular e imprimir el promedio de la clase* para

  Si *el contador no es igual a cero* **entonces**
  Establecer el promedio del total dividido por el contador
  Imprime el promedio
  Sino
  Imprima "No se ingresaron calificaciones"
  Fin Si

```
1   /* Fig. 3.8: fig03_08.c
2      Programa promedio de la clase con
3      repetición controlado por centinela */
4   #include <stdio.h>
5
6   int main()
7   {
8      float promedio;              /* nuevo tipo de dato */
9      int contador, nota, total;
10
11     /* fase inicialización */
12     total = 0;
13     contador = 0;
14
15     /* fase procesamiento */
16     printf( "Ingrese nota, -1 para terminar: " );
17     scanf( "%d", &nota );
18
19     while ( nota != -1 ) {
20        total = total + nota;
21        contador = contador + 1;
22        printf( "Ingrese nota, -1 para terminar: " );
23        scanf( "%d", &nota );
24     }
```



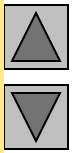
Outline

1. Iniciar las variables

2. Obtener la entrada del usuario

2.1 Realizar el bucle

```
25
26      /* fase de finalización */
27      if ( contador != 0 ) {
28         promedio = ( float ) total / contador;
29         printf( "Promedio de la clase es %.2f", promedio);
30      }
31      else
32         printf( "Ninguna nota fue ingresada\n" );
33
34      return 0;   /* indica finalización exitosa */
35   }
```



Outline

3. Calcular el promedio

3.1 Imprimir resultados

Salida del Programa

```
Ingrese nota, -1 para terminar: 75
Ingrese nota, -1 para terminar: 94
Ingrese nota, -1 para terminar: 97
Ingrese nota, -1 para terminar: 88
Ingrese nota, -1 para terminar: 70
Ingrese nota, -1 para terminar: 64
Ingrese nota, -1 para terminar: 83
Ingrese nota, -1 para terminar: 89
Ingrese nota, -1 para terminar: -1
Promedio de la clase es 82.50
```

3.10 Estructuras de Control Anidadas

- Problema
  - Una universidad tiene una lista de los resultados de los exámenes (1 = aprobado, 2 = reprobado) de 10 estudiantes.
  - Escriba un programa que analice los resultados
    - Si más de 8 estudiantes pasan, imprime "Aumentar la matrícula"
- Note que
  - El programa debe procesar 10 resultados de pruebas
    - Ciclo controlado por contador será utilizado
  - Dos contadores pueden ser utilizados
    - Uno por el número de pases, uno por el número de fallas
  - Cada resultado de la prueba es un número, ya sea un 1 o un 2
    - Si el número no es un 1, asumimos que es un 2

3.10 Estructuras de Control Anidadas (II)

- Esquema de nivel superior

  Analizar los resultados de los exámenes y decidir si se debe aumentar la matrícula

- Primer Refinamiento

  Iniciar las variables

  Introduce las diez notas de los exámenes y cuenta los aprobados y los fracasos.

  Imprimir un resumen de los resultados del examen y decidir si se debe aumentar la matrícula

- Refinar *Inicializar variables* para

  Iniciar pases a cero

  Iniciar las fallas a cero

  Iniciar el contador de estudiantes a uno

3.10Estructuras de Control Anidadas (III)

- Refina *la entrada de las diez calificaciones de los exámenes y cuenta los aprobados y los fracasos* para

```
Mientras que el contador de estudiantes es menor o igual a diez hacer
    Ingresa el resultado del próximo examen
    Si el estudiante pasó entonces
        Añade uno a los aprobados
    Sino
        Añade uno a los no aprobados
    Fin Si
    Añade uno al contador de estudiantes
Fin Mientras
```


3.10Estructuras de Control Anidadas (IV)

- Refinar *Imprimir un resumen de los resultados del examen y decidir si la matrícula debe ser elevada*

  Imprime el número de aprobados

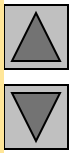
  Imprime el número de no aprobados

  `Si` *más de ocho estudiantes aprobaron* **`entonces`**

  Imprimir "Aumentar la matrícula"

  `Fin Si`

```c
/* Fig. 3.10: fig03_10.c
   Análisis de los resultados de los exámenes */
#include <stdio.h>

int main()
{
   /* inicializar las variables en las declaraciones */
   int passes = 0, failures = 0, student = 1, result;

/* procesar 10 estudiantes; bucle controlado por contador */
   while ( student <= 10 ) {
      printf("Ingrese resultado (1=aprobado,2=No aprobado ): "
      scanf( "%d", &result );

      if ( result == 1 )      /* SI/SINO aniddo en Mientras */
         passes = passes + 1;
      else
         failures = failures + 1;

      student = student + 1;
   }

   printf( "Aprobado %d\n", passes );
   printf( "No aprobado %d\n", failures );

   if ( passes > 8 )
      printf( "Aumentar Matricula\n" );

   return 0;   /* finalización exitosa */
}
```

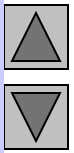


Outline

1. Iniciar las variables

2. Datos de entrada y recuento de aprobados/fracasos

3. Impresión de resultados



Outline

Salida del Programa

```
Ingrese Resultado (1=aprobado,2=No aprobado): 1
Ingrese Resultado (1=aprobado,2=No aprobado): 2
Ingrese Resultado (1=aprobado,2=No aprobado): 2
Ingrese Resultado (1=aprobado,2=No aprobado): 1
Ingrese Resultado (1=aprobado,2=No aprobado): 1
Ingrese Resultado (1=aprobado,2=No aprobado): 1
Ingrese Resultado (1=aprobado,2=No aprobado): 2
Ingrese Resultado (1=aprobado,2=No aprobado): 1
Ingrese Resultado (1=aprobado,2=No aprobado): 1
Ingrese Resultado (1=aprobado,2=No aprobado): 2
Aprobados 6
No aprobados 4
```

3.11 Operador de Asignamiento

- Los operadores de asignación abrevian las expresiones de asignación

  `c = c + 3;`

  puede abreviarse como `c += 3;` utilizando el operador de asignación de suma

- Declaraciones del formulario

  variable **=** *expresión de operador variable* **;**

  puede ser reescrito como

  *variable operator***=** *expression* **;**

- Ejemplos de otros operadores de asignación

```
d -= 4      (d = d - 4)
e *= 5      (e = e * 5)
f /= 3      (f = f / 3)
g %= 9      (g = g % 9)
```


3.12 Operadores Incremental y Decremental

- Operador Incremental **(++)** - puede ser utilizado en vez de **c+=1**
- Operador Decremental **(--)** - puede ser utilizado en vez de **c-=1**.
- Pre incremento
  - Operador es usado antes de la variable (**++c** o **--c**)
  - Variable es modificada, entonces la expresión es ejecutada
- Post incremento
  - Operador es usado después de la variable (**c++** o **c--**)
  - La expresión se ejecuta, luego la variable se cambia
- Si **c = 5**, entonces

```
printf( "%d", ++c);
```

    - Imprime **6**

```
printf( "%d", c++);
```

    - Imprime **5**

  - En cualquier caso, **c** tiene el valor de **6**

3.12 Operadores Incremental y Decremental (II)

- Cuando la variable no está en una expresión
  - La pre incrementación y la post incrementación tienen el mismo efecto.

```
++c;
printf( "%d", c);
```

y

```
c++;
printf( "%d", c);
```

tienen el mismo efecto.