Note Title cout — Imprime algo — 8/29/2011

Requiere de la librería <iostream>:

```
#include <iostream>
using namespace std;
```

Sintaxis: `cout << ____;`
`cout << ____ << ____;`

a) Un letrero: Quiero que salga tal cual en pant. Uso " ".

Ej:
```
cout << "Hola, Juan";
```
ó → Hola, Juan

```
cout << "Hola, " << "Juan";
```
ó
```
cout << "Hola, ";
cout << "Juan";
```

b) Un letrero de una sola letra: Uso ' '

Ej:
```
cout << 'B';
```
→ B

c) El valor almacenado en una var:
- No se usan comillas
- La var. debe estar previamente definida y debe contener algún valor.

Ej:
```
double a;
a = 3.79;
cout << a;
```
(caso (c))

mem: 3.79 | a | double

pantalla: 3.79

Ej:
```
cout << 'a';
```
(caso (b))

pantalla: a

d) El resultado de una operación matemática:
- No lleva comillas

Ej:
```
double x = 3.1;
double y = 4.2;
cout << x+y+0.1;
```
memoria: 3.1 | x | double ; 4.2 | y | double

pantalla: 7.4

e) Caracteres especiales como el enter y el tab:
- Van adentro de comillas como si fueran letreros, PERO no se imprimen tal cual (excepción)
- Están precedidos de \

Ej:
```
cout << '\n';
```
(1 caracter)

→ (salto de línea)

⇓

"Escape Sequence"

2 caracteres

cout << "\n\n";

\n → enter
\t → tab
\\ → \
\" → "
\' → '

cout << "Hola,\nJuan\n";

Hola,
Juan

cout << "Hola," << '\n' << "Juan\n";

cout << "Hola,";
cout << "\nJuan\n";

f) Una combinación de los casos anteriores.

Ej: double m=3.5, n=4.2;

memoria

3.5	4.2
m	n
double	double

cout << "Hola, Juan:\n\n"
<< "mira la sigt. op.: "
<< '\t' << "m+n+2.1=" << m+n+2.1 << "\n¿Que"
<< " tal?" << "\n\n\n"
<< "Di:\"Sol\"\n\n";

pantalla

Hola, Juan:

mira la sigt. op.: m+n+2.1=9.8
¿Que tal?

Di:"Sol"

cin — Lee una información que se entra por el teclado.

Actúa así (en tiempo de ejecución):
1. Detiene momentáneamente la ejecución del prog. para esperar por un dato que debe entrar el usuario (activa el teclado)
2. Cuando el usuario entra el dato, y le da a la tecla de enter (indicando que terminó de entrar su info.), el dato se lee y se almacena en la var. que acompaña al cin.

*NOTA: Como el teclado está activo, el enter del usuario hace que el cursor baje a la próxima línea en la pantalla.

Requiere de la librería <iostream>:

```
#include <iostream>
using namespace std;
```

Sintaxis:

```
cin >> l;        cin >> l >> an;
```

nombre de la var. que recibe el valor q. se entra.

NOTA: Se suele preceder toda instrucción cin, por un cout que especifica al usuario cuál es el dato que el prog. espera que le provean en ese momento.

```
double l, an;
cout << "Entre el largo del rect.: ";
cin >> l;
cout << "Entre el ancho del rect.: ";
cin >> an;
```

l: 10.3 — an: 3.8

Entre el largo del rect.: 10.3
Entre el ancho del rect.: 3.8

```
cout << "Entre el largo y el ancho del rect.: ";
cin >> l >> an;
```

10.3 3.8

```
cin >> l;
cin >> an;
```

Entre el largo y el ancho del rect.:
10.3
3.8

A = l * an;

Operación de Asignación (=)

Significa: Resuelve el lado derecho del signo de = y el resultado guárdalo ó almacénalo en la var. de la izq. del signo de =.

Ej:

```
int i;                  i = 9;
double A, B=3.1, C=2.0; A = B + C;
double x, y = 1.3;      x = y - 2.5;
char C1, C2;            C1 = 'A'; C2 = C1;
```

Operadores Matemáticos

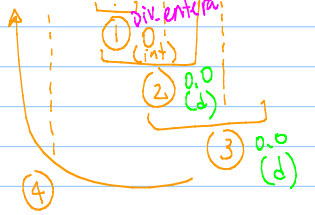
		Matem.	C++
a)	Suma	+	+
b)	Resta	−	−
c)	Multiplicación	x , ·	*
d)	División ó Div. Entera	÷ , — , /	/
e)	Módulo (residuo)	MOD	%

int int → 9 / 2 — Div. entera — Resp. 4 (int)

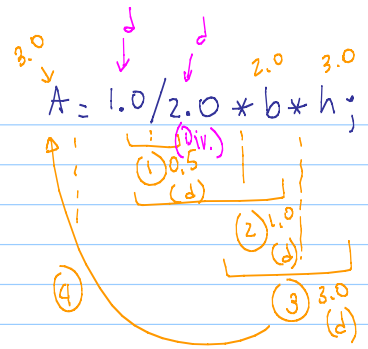
d int → 9.0 / 2 ó int d → 9 / 2.0 ó d d → 9.0 / 2.0 — División — 4.5 (double)

En C++, el tipo de dato de las vars. involucradas en la operación de div. (numerador y denom.), determina si la operación es div. ó div. entera.

Ej: `A = 1/2 * b * h;` // $A = \frac{1}{2}bh$



Ej: `A = 1.0/2.0 * b * h;`



El Modulo (%) me da el residuo, remanente o residual que resulta de dividir 2 #'s ENTEROS.

Por def., en C++, el % es UNA OPERACIÓN DE DATOS ENTEROS.

9/2

$\frac{9}{2}$

```
   4
2 | 9
   -8
   1   ← 9%2
```

9/5 → 1

$\frac{9}{5}$

9%5 → 4

```
   1   ← 9/5
5 | 9
   -5
   4   ← 9%5
```

$\frac{2}{9}$ 2/9 → 0

2%9 → 2

```
   0
9 | 2
   -0
   2
```

3%4 → 3
4%4 → 0
5%4 → 1
6%4 → 2
7%4 → 3
8%4 → 0
9%4 → 1
10%4 → 2
11%4 → 3
12%4 → 0
13%4 → 1
14%4 → 2
15%4 → 3
16%4 → 0

```
   0
4 | 3
   -0
   3
```

16.0%5.0 → ERROR DE COMPILACIÓN

(*) Solo se permite int % int, y la resp. es 1 int.

int + int → int
int - int → int
int * int → int
int / int → int
↖ Div. entera
int % int → int

int + d ó d + int ó d + d → d
int - d d - int d - d → d
int * d d * int d * d → d
int / d d / int d / d → d
↖ Div. ↖ Div ↖ Div
int % d d % int d % d → ERROR DE COMPILAC.

Jerarquía de Operadores.

Es el orden en que C++ ejecuta varias operaciones que aparecen juntas en una misma línea de código o instrucción.

SIEMPRE C++ selecciona una operación por vez, siguiendo el orden de la jerarquía. (más arriba ⇒ mayor prioridad)

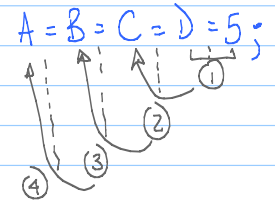
Cuando aparecen, en una misma instrucción, 2 operaciones con la misma prioridad, se selecciona una a la vez, de acuerdo a un 2do criterio, que usualmente es de izq. a der.

↑ prioridad

1. ( ) → conversiones
2. *, /, %
3. +, −
4. = (operación de asignación). → Der. a Izq.

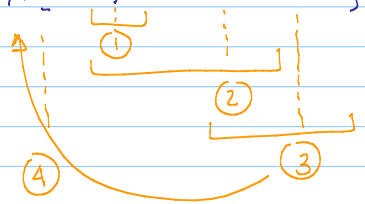
De izq. a Der.

↓ prioridad

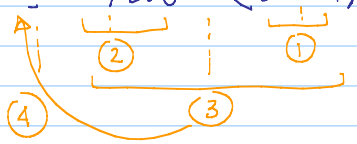


Área del Δ: $A = \frac{1}{2}bh$ (matem.)

Ej: `A = 1.0/2.0 * b * h;`



Ej: `A = 1.0/2.0 * (b*h);`

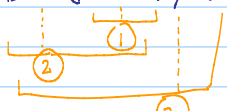


1. ( )
2. *, /, %
3. +, −
4. =

Ej: $A = \frac{B(C+5)}{3C+2B}$ (matem)

$A = \frac{1}{2}bh$

Sol. 1: (ERRONEA) `A = B*(C+5)/3*C+2*B;` ERROR LÓGICO pues lo que se programó no es lo que se quiere sino ↯



$A = \frac{B(C+5)}{3}.C + 2B$

Sol. 2: (bien) `A = B*(C+5)/(3*C+2*B);` OK

Tres puntos a considerar al escribir una fórmula computacional, para estar seguros de que está bien:

1) Use los operadores que C++ entiende

Ej: * para mult.

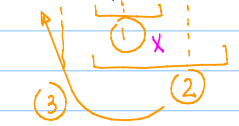
`a x b` → mal (ERROR DE COMP.)

`a *b` → bien

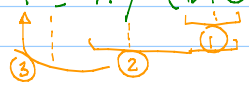
2) Aplique la jerarquía de operadores de C++ y verifique que el orden en que se evalúan las operaciones está bien.

Ej: $F = \frac{A}{B+C}$ → fórmula matem.

`F = A/B+C;` → mal (ERROR LÓGICO)



`F = A/(B+C);` → bien



3) Verifique el tipo de dato de los operandos envueltos en cada paso, sobre todo (ó particularmente) cuando tiene divisiones, divisiones enteras y/o módulos (%).

Ej: `A = 1/2 * b * h;` (1/2: int int) → ERROR LÓGICO (siempre da resp. 0.0)

(1: int, 2: int; / : X (Div. entera))

`F = C % D;` (C: int, D: d) → ERROR DE COMPILACIÓN

Resumen

int + int	→	int
int - int	→	int
int * int	→	int
int / int	→	int
int % int	→	int

(/ : Div. entera)

int + d	ó	d + int	ó	d + d → d
int - d		d - int		d - d → d
int * d		d * int		d * d → d
int / d		d / int		d / d → d
int % d		d % int		d % d → ERROR DE COMPILACIÓN

(/ : División)

Note Title ASCII 8/17/2011

American Standard Code for Information Interchange

'0'	
'1'	
'2'	
'A'	65
'B'	66
'C'	67
⋮	
'Z'	
⋮	
'a'	7?
'b'	
'c'	
⋮	

La tabla ASCII contiene todos los caracteres y el código numérico único que se asocia con cada caracter.

Buscar y fotocopiar la tabla ASCII.

'é'	130
⋮	
'á'	160
'í'	161
'ó'	162
'ú'	163
'ñ'	164
'Ñ'	165
⋮	

La tabla básica tiene 128 caracteres y la extendida tiene 128 más.

Total: 256 caracteres

Note Title 8/29/2011

Constante

const double PI = 3.1416;

(PI: 3.1416, double, const)

PI = 29.8; ✗ ERROR DE COMPILACIÓN

variable

```
int s;
s = 10;
{
s = 15;
}
s = -9;
```

int s = 10; (s: ~~15~~ ~~10~~ -9, int)

double PI = 3.1416;

```
double PI;   //variable
PI = 3.1416;
PI = 18.9;
```

(PI: ~~3.1416~~ 18.9, double)

Conversiones

Requisito: Si necesita convertir de un tipo de dato a otro, SIEMPRE hágalo de forma EXPLÍCITA.

Caso 1: ¿Qué pasa si un resultado int lo guardo en una var. real (d)? Conversión de int a d

```
int A = 9, B = 2;
double R;
```

FORMA IMPLÍCITA:

R = A / B; (A = 9, B = 2 → ① 4 (int) (Div. entera) → ② R = 4.0)

R: 4.0, double

FORMA EXPLÍCITA:

R = (double)(A / B); (① 4 (int) (Div. entera) → ② 4.0 (d) → ③ R = 4.0)

Resp.: Se convierte a real; es decir, se presume que los decimales son 0

Caso 2: ¿Qué pasa si un resultado real (double) se almacena en una variable entera (int)? Conversión de double a int

```
double x = 9.0, y = 2.0;
int R1;
```

FORMA IMPLÍCITA (WARNING):

R1 = x / y; (x = 9.0, y = 2.0 → ① 4.5 (Div.) (d) → ② R1 = 4)

R1: 4, int

FORMA EXPLÍCITA:

R1 = (int)(x / y); (① 4.5 (d) (Div.) → ② 4 (int) → ③ R1 = 4)

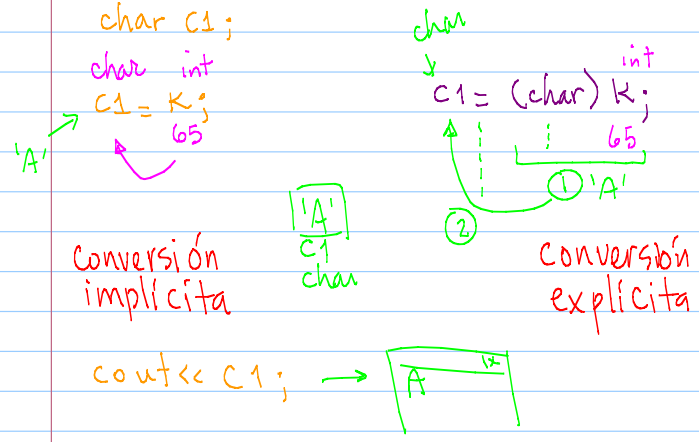
ASCII

Resp: Se almacena lo que se puede, que es la parte entera, y se pierden los decimales.

Caso 3: ¿Qué sucede si un valor entero se guarda en una variable char?

Conversión de int a char

```
int K=65;
char C1;
C1=K;
```



Conversión implícita

Conversión explícita

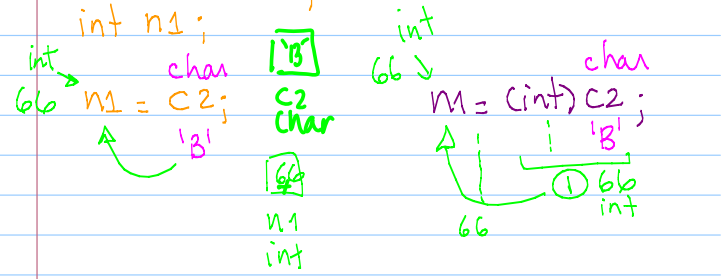
```
cout<< C1;
```

→ A

Resp: Se almacena la letra ó caracter correspondiente a ese # en la tabla ASCII.

Caso 4: ¿Qué sucede si un valor char se guarda en una variable entera? Conversión de char a int.

```
char C2='B';
int n1;
n1=C2;
```



Conversión implícita

Conversión explícita

Resp: Se almacena el # entero asociado con ese caracter en la tabla ASCII.

Caso 5 (*): Cuando, en una var. bool se entra un dato que no es bool (ni 1 ni 0), se interpreta así: TODO LO QUE NO SEA 0, ES 1 (true). (Conversión de un tipo de dato cualquiera a bool).

Caso 6 (*): Si almacena un true en una var. int, se almacena un 1; y si almacena un false, se almacena un cero (0). Conversión de bool a int.

* Siempre que se necesite hacer una conversión en un programa se debe hacer de forma explícita, así evitamos un posible warning (alerta) en la compilación.

```cpp
#include <iostream>
using namespace std;

int main()
{
	int i1=5, i2=66, i3=79;
	double d1=3.99, d2=5.99, d3=9.0, d4=2.0;
	char c1='A', c2='M';
	bool l1=true;

	cout<<"i1="<<i1<<", i2="<<i2<<", i3="<<i3<<'\n';
	cout<<"d1="<<d1<<", d2="<<d2<<", d3="<<d3<<", d4="<<d4<<'\n';
	cout<<"c1="<<c1<<", c2="<<c2<<'\n';
	cout<<"l1="<<l1<<'\n';

	//conversion implicitade int a double
	d1=i1; //d1 recibe 5 y le anade .0 a los decimales = 5.0
	cout<<"d1="<<d1<<'\n';

	//conversion implicita de double a int
	i3=d2; //produce warning
	cout<<"i3="<<i3<<'\n';
	//de forma explicita no hay warning
	i3=(int)d2;
	cout<<"i3="<<i3<<'\n';

	//conversion implicita de int a char
	c1=i2; // la conversion explicita
	       //(recomendada) seria c1=(char)i2;
	cout<<"c1="<<c1<<", i2="<<i2<<'\n';

	//conversion implicita de char a int
	i3=c2; //la forma explicita (recomendada) es i3=(int)c2;
	cout<<"i3="<<i3<<", c2="<<c2<<'\n';

	c2=c1; //no hay conversion
	cout<<"c2="<<c2<<'\n';

	cout<<"d3/d4="<<d3/d4<<'\n';
	cout<<"(int)d3/(int)d4="<<(int)d3/(int)d4<<'\n';

	cout<<9.0/2.0<<'\n';
	cout<<9/2<<'\n';
	cout<<9%2<<'\n';

	cout<<"Entra una letra: ";
	cin>>c1;

	cout<<"\nEl numero correspondiente a la letra "<<c1
	      <<" es "<<(int)c1<<"\n\n";

	cout<<"Entra un numero: ";
	cin>>i1;

	cout<<"Tu numero es "<<i1<<"\n\n";
	return 0;
}
```

```
i1=5, i2=66, i3=79
d1=3.99, d2=5.99, d3=9, d4=2
c1=A, c2=M
l1=1
d1=5
i3=5
i3=5
c1=B, i2=66
i3=77, c2=M
c2=B
d3/d4=4.5
(int)d3/(int)d4=4
4.5
4
1
Entra una letra: 1

El numero correspondiente a la letra 1 es 49

Entra un numero: 1
Tu numero es 1

Press any key to continue . . .
```

en C++

Escribe una fórmula computacional que se corresponda con las sigts. fórmulas matemáticas.

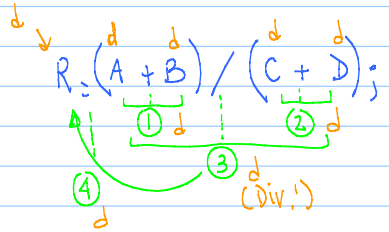
Presunciones para este ejercicio

- Si no me dicen el tipo de dato de las vars, presumo double.
- Si no me dicen si es div. ó div. entera, presumo división

a) $R = \frac{A+B}{C+D}$

```
R = A + B / C + D;
```

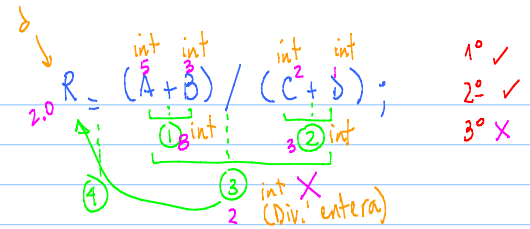
1º ✓ operadores
2º ✗ jerarquía

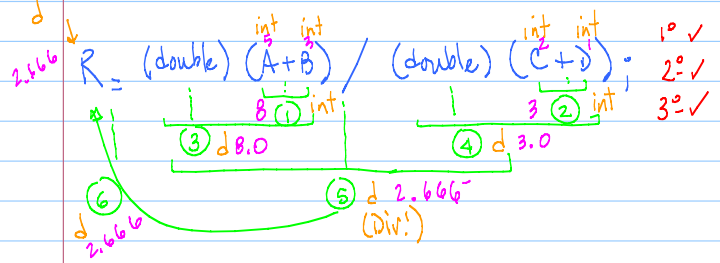


b) La misma fórmula de (a), pero presuma que A, B, C, D son enteras.

$$R = \frac{A+B}{C+D}$$

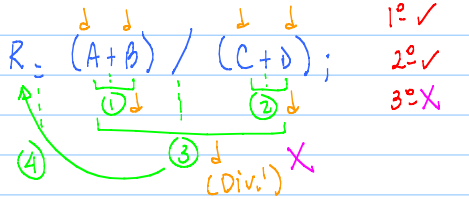
A=5, B=3, C=2, D=1

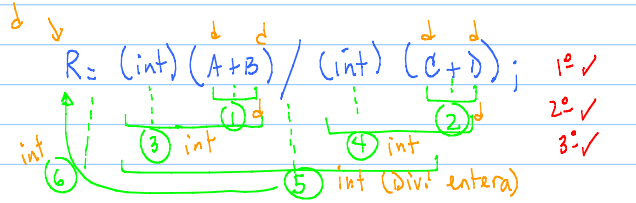




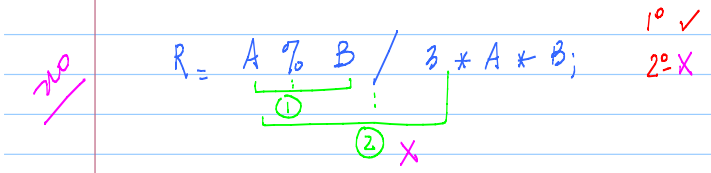
c) La misma fórmula que en (a), pero se quiere div.' entera (nota: Las vars. son double).

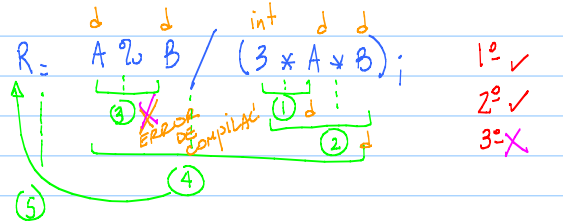
$$R = \frac{A+B}{C+D}$$



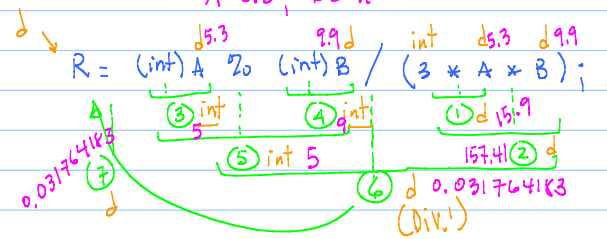


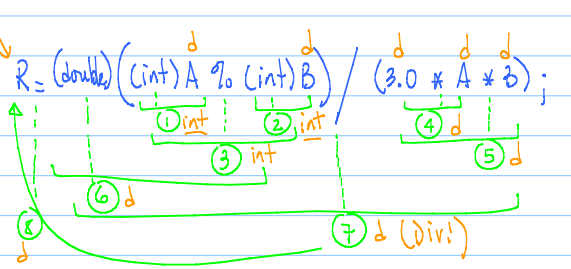
d) $R = \frac{A \bmod B}{3AB}$



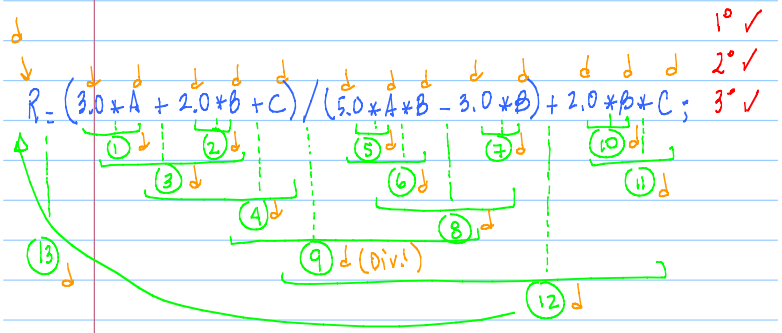


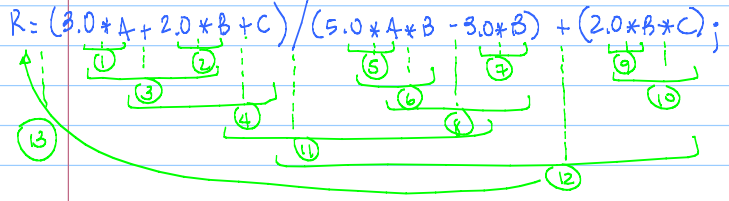
A=5.3, B=9.9





e) $R = \frac{3A+2B+C}{5AB-3B} + 2BC$



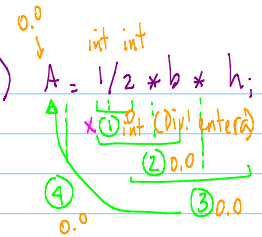


Ejercicio: La fórmula matemática para calcular el área de un Δ es: $A = \frac{1}{2}bh$. Conteste lo sigt:

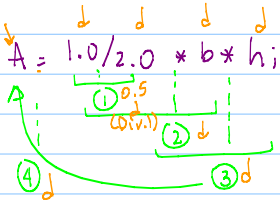
① ¿Cuáles de las sigts fórmulas computacionales representan adecuadamente en C++ la fórmula del cálculo del área del Δ? (circule todas las que apliquen).

② ¿Cuáles se consideran las mejores representaciones computacionales? (coloque ⊛ al lado de todas las que apliquen)

a) `A = 1/2 * b * h;` — 1° ✓ 2° ✓ 3° X

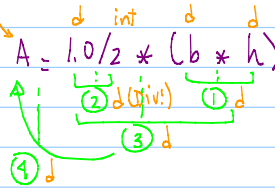


⊛ b) `A = 1.0/2.0 * b * h;` — 1° ✓ 2° ✓ 3° ✓

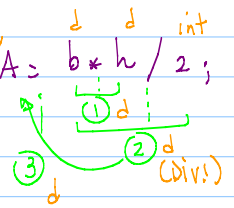


c) `A = 1.0/2.0 (b * h);` — 1° X

d) `A = 1.0/2 * (b * h);` — 1° ✓ 2° ✓ 3° ✓



e) `A = b * h / 2;` — 1° ✓ 2° ✓ 3° ✓



(*) f) A = 0.5 * b * h; 1° ✓ 2° ✓ 3° ✓

(*) g) A = (b * h) / 2.0; 1° ✓ 2° ✓ 3° ✓

h) A = b * (h/2); 1° ✓ 2° ✓ 3° ✓

i) A = b (h/2); 1° ✗

Operadores Relacionales

- operandos: del mismo tipo
- resultado: bool (T/F)

Se usan para programar preguntas o decisiones. Comparan datos. Su resp. es siempre bool (T/F).

```
int A=9, B=2;
```

		Matem.	C++		
a)	Mayor que	>	>	A > B (9 > 2)	true
b)	Menor que	<	<	A < B	false
c)	Mayor o igual que	≥	>=	A >= B	true
d)	Menor o Igual que	≤	<=	A <= B	false
e)	Igual	=	==	A == B	false
f)	Diferente	<>, ≠	!=	A != B	true

```
char C1='A', C2='B';
```

C1 (char) 'A' 65 | C2 (char) 'B' 66

C1 > C2 → false

C1 ('A' 65) > 'q' (97) → false

c1 > "Amor" false
'A' ↑
65 65

"Rosana" > "Juan" true
↑ ↑

string char
"10" > '2' false
↑ ↑
49 50

int int
10 > 2 true

"Buen"
s1
string

"Buen"
s1 > "Pedro" false

Observe la diferencia entre la comparación de textos y la comparación de números.

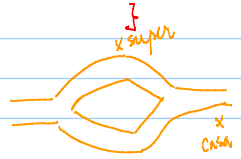
Instrucción if

condición ó pregunta
Resp: (T/F) bool

```
if ( _____ )
{
    ___;
}else
{
    ___;
}
```

T
F

Si (condición evalúa a T)
{
haz esto
} sino
{
haz esto otro
}



Ejercicio:

- Si las horas trabajadas en una semana por 1 empl. son mayores que 40 entonces imprima "trabajó horas extras" ; sino imprima "no trabajó horas extras".

```
double hrs;

cout<< "Entre las horas trabajadas en la"
    << "semana por el empleado: ";
cin >> hrs;

if( hrs > 40.0 )
{
    cout<< "Trabajó horas extras\n\n";
}else
{
    cout<< "No trabajó horas extras\n\n";
}
```

T
F

ó

```
if ( hrs <= 40.0 )
{
    cout<<"No trabajo horas extras\n\n";
} else
{
    cout<<"Trabajo horas extras\n\n";
}
```

T

F

PROBLEMA:

Una compañía paga a sus empleados por las horas trabajadas en la semana. Las horas extras las paga al doble del pago por hora regular. Escriba un prog. en C++ que permita calcular e imprimir la cantidad a pagar en una semana a un empleado que cobra por horas.

(*) Una jornada de trabajo regular tiene 40 horas a la semana.

FASE 1: Análisis

Entradas	Procesamiento	Salidas
• Pago por hora del empleado (pph) • Horas trabajadas por el empleado en esa sem. (hrs)	1) Solicitar el pago por hora del empl (pph) y las horas que trabajó en esa semana (hrs) y validar 2) Si el empleado trabajó más de 40 horas en la sem (hrs > 40): a) Calcular las horas extras que trabajó (he): he = hrs - 40 b) Calcular la cant. a pagar al empl. esta semana (total): total = 40 × pph + he × pph × 2	• Cantidad a pagar a un empleado en una semana (total)

3) Si el empl. no trabajó más de 40 horas en la semana:

a) Calcular el pago de la semana del empl. (total):

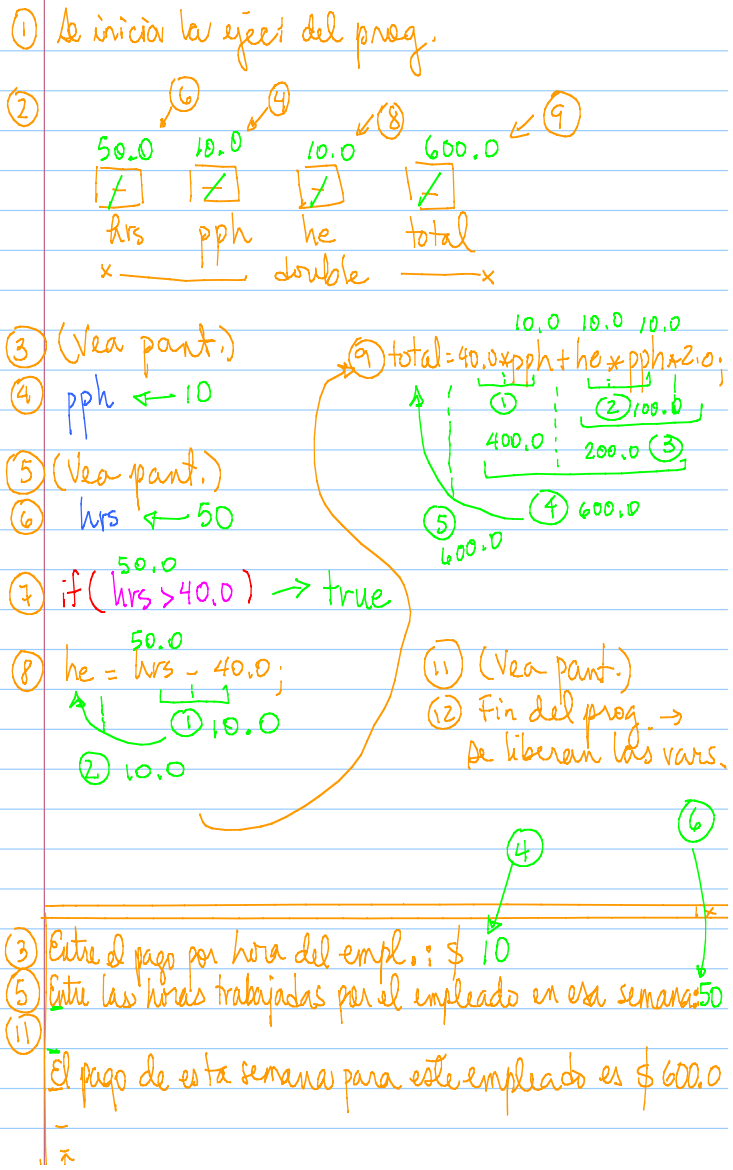
total = hrs × pph

4) Imprimir el pago de una semana del empleado (total)

FASE 2: PROGRAMA

```
#include <iostream>
using namespace std;
(1) int main()
{
  (2) double hrs, pph, he, total;

  (3) cout<< "Entre el pago por hora del empl.: $";
  (4) cin>> pph;

  (5) cout<< "Entre las horas trabajadas por"
        <<" el empleado en esa semana: ";
  (6) cin >> hrs;

  (7) if( hrs > 40.0)
  T [ {
      (8) he = hrs - 40.0;
      (9) total = 40.0 * pph + he * pph * 2.0;
    }else
  F [ {
      (10) total = hrs * pph;
    }

  (11) cout<< "\nEl pago de esta semana para"
        <<" este empleado es $" << total<< "\n\n";

  (12) return 0;
}
```

FASE 3: TEST ó PRUEBA MANUAL DE LA SOL.



Validación de los datos de entrada:

Es el proceso de verificar, en la medida de lo posible, que un dato que se solicita al usuario, se entre correctamente antes de proceder a trabajar con él en la solución del problema.

Ej:
- Un precio debe ser cant. positiva ó cero (gratis!)
- Cant. de estudiantes no debe ser negativo.
- Una nota debe ser una cant. entre 0 y 100 etc.

Si al ejercicio anterior se le incorpora la validación de los datos de entrada (o sea, la verificación, antes de proceder a solucionar el problema, de que las horas trabajadas por el empleado en la semana y su pago por hora NO sean negativos)

De ahora en adelante, todo dato de entrada que sea validable, se debe validar.

En la próxima pág. se presenta el ejercicio de calcular el pago semanal para un empleado que trabaja por horas, que ya se solucionó en la clase anterior, pero AHORA INCLUYE LA VALIDACIÓN de los datos de entrada

```cpp
#include "stdafx.h"
using namespace System;

#include <iostream>
using namespace std;

int main()
{
	double hrs, pph, he, total;

	cout<<"Entre el pago por hora del empleado: $";
	cin>>pph;

	if(pph < 0.0)
	{
		cout<<"\nEl pago por hora no debe ser negativo\n\n";
	}else
	{
		cout<<"Entre las horas trabajadas por el empleado en la "
		    <<"semana: ";
		cin>>hrs;

		if(hrs < 0.0)
		{
			cout<<"\nHoras trabajadas invalidas, no deben ser "
			    <<"negativas\n\n";
		}else
		{
			if(hrs > 40.0)
			{
				he = hrs - 40.0;
				total = 40.0 * pph + he * pph * 2.0;
			}else
			{
				total = hrs * pph;
			}

			cout<<"\nEl pago de esta semana para el empleado "
			    <<"es $"<<total<<"\n\n";
		}
	}
	return 0;
}
```

Prueba Manual pph -25

① Se inicia la ejecución del programa

②

④ -25.0

hrs	pph	he	total
-	/	-	-
x	double		x

③ (Vea pantalla)

④ pph ← -25

-25.0

⑤ if (pph < 0.0) → True

⑥ (Vea pant.)

⑯ Fin del prog. → Se liberan las variables

④

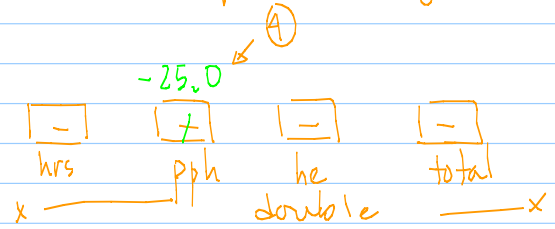
③ Entre el pago por hora del empleado: $ -25

⑥ El pago por hora no debe ser negativo

Test Manual pph -25

① Se inicia la ejec. del prog.

②



③ (Vea pantalla)

④ pph ← -25

⑤ if (pph < 0.0) → true
-25.0

⑥ (Vea pant.)

⑯ Fin del prog. → Se liberan las vars. ④

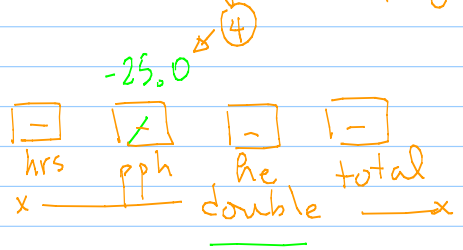
③ Entre el pago por hora del empleado: $ -25

⑥ El pago por hora no debe ser negativo

Prueba pph -25.0

① Se inicia la ejec. del prog!

②



③ (Vea pant.)

④ pph ← -25

⑤ if (pph < 0.0) → true
-25.0

⑥ (Vea pant.)

⑮ Fin del programa → Se liberan las variables.

③ Entre el pago por hora del empleado: $ -25

⑥ El pago por hora del empleado no debe ser negativo

Tarea

① Prueba Manual para pph 25
hrs 30

② Prueba Manual para pph 25
hrs -30