# FUNDAMENTOS DE PROGRAMACIÓN

Inicio Clase 03

Profesor: Carlos Díaz

# Clase 03: Asignación, formato y funciones matemáticas

- Operaciones de asignación =
- Operaciones de acumulación
- Formato de salida
- Funciones matemáticas

# Operaciones de asignación =

```
factor = 1.06;
peso = 155.0;
pesoTotal = factor * peso;


c = 25;
b = 25;
a = 25;

a = b = c = 25;
```

# Operaciones de acumulación

```
suma = suma + 10;
suma += 10;

precio = precio * tasa;
precio *= tasa;

precio = precio * (tasa + 1);
precio * = tasa + 1;
```

+=  −=  *=  /=  %=

# Conteo

```
i = i + 1;

n = n - 1;

j = j + 2;

m = m - 2;
```

| Expresión | Alternativa |
|---|---|
| `i = i + 1` | `i++ o ++i` |
| `n = n + 1` | `n++ o ++n` |
| `contador = contador + 1` | `contador++ o ++contador` |
| `i = i - 1` | `i-- o --i` |
| `n = n - 1` | `n-- o --n` |
| `contador = contador - 1` | `contador-- o --contador` |

# Ejemplo

```
#include <iostream>
using namespace std;
int main()
{
    int n=8,k;
    k=++n;
    cout<<"n= "<<n<<" k= "<<k<<endl; //Muestra 9 y 9
    n=8;
    k=n++;
    cout<<"n= "<<n<<" k= "<<k<<endl; //Muestra 9 y 8
    return 0;
}
```

# Formato de salida de números

- El formato de números desplegado por cout puede controlarse mediante manipuladores. Si el manipulador requiere un argumento debe incluirse la librería <iomanip>. Presentamos los manipuladores más comunes:

```
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{
        cout<<8<<endl; //Uso de setw(n)
        cout<<985<<endl;
        cout<<setw(8)<<8<<endl;
        cout<<setw(8)<<985<<endl;
        cout<<985.0/8<<endl; //Uso de fixed
        cout<<fixed<<985.0/8<<endl;
        return 0;
}
```

| Manipulador | Acción |
|---|---|
| setw(n) | Establece el ancho del campo en n. |
| fixed | Se muestra un punto decimal y usa seis dígitos por omisión después del punto decimal. Rellena con ceros a la derecha si es necesario. |

# Formato de salida de números

- El formato de números desplegado por cout puede controlarse mediante manipuladores. Si el manipulador requiere un argumento debe incluirse la librería <iomanip>. Presentamos los manipuladores más comunes:

```
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{
        cout<<8<<endl; //Uso de setw(n)
        cout<<985<<endl;
        cout<<setw(8)<<8<<endl;
        cout<<setw(8)<<985<<endl;
        cout<<985.0/8<<endl; //Uso de fixed
        cout<<fixed<<985.0/8<<endl;
        return 0;
}
```

| Manipulador | Acción |
|---|---|
| setw(n) | Establece el ancho del campo en n. |
| fixed | Se muestra un punto decimal y usa seis dígitos por omisión después del punto decimal. Rellena con ceros a la derecha si es necesario. |

# Formato de salida de números

```
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{
        //Uso de setfill('carácter')
        cout<<setw(6)<<setfill('*')<<12.3<<endl;
        //Uso de setprecision(n)
        cout<<1024.0/7<<endl;
        cout<<setprecision(10)<<1024.0/7<<endl;
        cout<<fixed<<setprecision(10)<<1024.0/7<<endl;
        return 0;
}
```

| Manipulador | Acción |
|---|---|
| setfill('carácter') | Establece el carácter de relleno a la izquierda. Por defecto es un espacio. |
| setprecision(n) | Muestra n cifras significativas. Si se designa antes fixed, n será el número de decimales. |

# Formato de salida de números

```
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{
        //Uso de scientific
        cout<<scientific<<123.45<<endl;
        cout<<fixed<<setprecision(2)<<scientific<<123.45<<endl;
        cout<<oct<<15<<endl; //Uso de showbase
        cout<<hex<<15<<endl;
        cout<<dec<<0xF<<endl;
        cout<<showbase<<oct<<15<<endl;
        cout<<showbase<<hex<<15<<endl;
        cout<<showbase<<dec<<0xF<<endl;
        return 0;
}
```

| Manipulador | Acción |
|---|---|
| scientific | Muestra los números en notación científica. |
| oct<br>hex<br>dec | Muestra en la base indicada. |
| showbase | Muestra la base usada. 0 a la izquierda es para números octales y 0x es para hexadecimales. |

# Formato de salida de números

```
#include <iostream>
using namespace std;
int main()
{
        //Uso de boolalpha
        bool respuesta=true;
        cout<<respuesta<<endl;
        cout<<boolalpha<<respuesta<<endl;
        //Uso de showpoint
        cout<<123.4<<endl;
        cout<<showpoint<<123.4<<endl;
        //Uso de showpost
        cout<<showpos<<123.4<<endl;
        return 0;
}
```

| Manipulador | Acción |
|---|---|
| showpoint | Siempre muestra seis dígitos en total y rellena con ceros a la derecha si es necesario. Para valores mayores se muestra en notación científica. |
| showpos | Muestra el signo + en todos los números positivos. |
| boolalpha | Muestra los valores lógicos como verdadero y falso, en lugar de 1 y 0. |

# Formato de salida de números

| Manipulador | Acción |
|---|---|
| left | Justifica a la izquierda todos los números. |
| right | Justifica a la derecha todos los números. |
| uppercase | Muestra dígitos hexadecimales y el exponente en notación científica en mayúsculas. |
| noboolalpha | Muestra los valores booleanos como 1 y 0, en lugar de verdadero y falso. |
| noshowbase | No muestra los números octales con 0 a la izquierda ni los hexadecimales con 0x a la izquierda. |
| noshowpoint | No usa punto decimal para reales sin parte decimal, ni rellena con ceros a la derecha y muestra un máximo de seis dígitos decimales. |
| noshowpos | No muestra el signo + a la izquierda de un número positivo. |
| nouppercase | Muestra dígitos hexadecimales y el exponente en notación científica en minúsculas. |

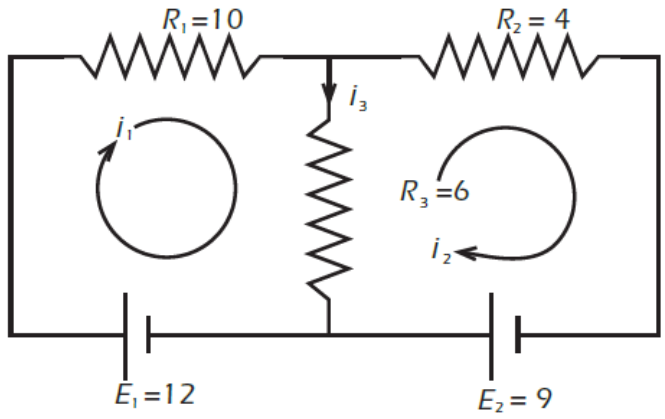
# Ejercicio 1

Para el circuito eléctrico mostrado en la figura , las corrientes en los ramales, $i_1$, $i_2$ e $i_3$ pueden determinarse usando las fórmulas

$$i_1 = \frac{E_2R_3 + E_1(R_1 + R_3)}{(R_1 + R_3)(R_2 + R_3) - (R_3)^2}$$

$$i_2 = \frac{E_1R_3 + E_2(R_1 + R_3)}{(R_1 + R_3)(R_2 + R_3) - (R_3)^2}$$

$$i_3 = i_1 - i_2$$




Usando estas fórmulas, escriba un programa en C++ para calcular las corrientes en los ramales cuando $R_1$ = 10 ohmios, $R_2$ = 4 ohmios, $R_3$ = 6 ohmios, $E_1$ = 12 voltios y $E_2$ = 9 voltios. El despliegue producido por su programa deberá ser

```
La corriente en el ramal 1 es xx.xxxxx
La corriente en el ramal 2 es xx.xxxxx
La corriente en el ramal 3 es xx.xxxxx
```

donde xx.xxxxx denota que el valor calculado deberá colocarse en un ancho de campo suficiente para dos lugares a la izquierda del punto decimal y cinco lugares a la derecha de éste.

# Funciones matemáticas

- Se debe incluir la librería <cmath>. A continuación algunas funciones:

| Nombre de la función | Descripción | Valor devuelto |
|---|---|---|
| `abs(a)` | valor absoluto | mismo tipo de datos que el argumento |
| `pow(a1,a2)` | a1 elevado a la potencia a2 | tipo de datos del argumento a1 |
| `sqrt(a)` | raíz cuadrada de un número real | precisión doble |
| `sin(a)` | seno de a (a en radianes) | doble |
| `cos(a)` | coseno de a (a en radianes) | doble |
| `tan(a)` | tangente de a (a en radianes) | doble |
| `log(a)` | logaritmo natural de a | doble |
| `log10(a)` | logaritmo común (base 10) de a | doble |
| `exp(a)` | e elevado a la potencia a | doble |

- **Tarea: Averiguar la lista completa de funciones de la librería cmath.**

# Ejemplo

El programa ilustra el uso de la función `sqrt` para determinar el tiempo que tarda una pelota en golpear el suelo después de haber sido dejada caer desde una torre de 800 pies. La fórmula matemática usada para calcular el tiempo, en segundos, que tarda en caer una distancia determinada, en pies, es

```
tiempo = sqrt(2 * distancia / g)
```

donde *g* es la constante gravitacional igual a 32.2 pies/s$^2$.

```
#include <iostream>
#include <cmath>
using namespace std;
int main()
{
        int altura;
        double tiempo;
        altura=800;
        tiempo=sqrt(2*altura/32.2);
        cout<<"Tardar\xA0 " << tiempo << " segundos en caer "
        <<altura << " pies.\n";
        return 0;
}
```

# Ejercicio 2

Las coordenadas polares de un punto consisten en la distancia, $r$, de un origen especificado y un ángulo, $\theta$, con respecto al eje $x$. Las coordenadas ($x$ y $y$) del punto se relacionan con sus coordenadas polares por las fórmulas

$$x = r \cos \theta$$
$$y = r \operatorname{seno} \theta$$

Usando estas fórmulas, escriba un programa en C++ que calcule las coordenadas ($x$, $y$) del punto cuyas coordenadas polares son $r = 10$ y $\theta = 30$ grados. Verifique los resultados producidos por su programa calculando los resultados en forma manual. Después de haber verificado que su programa funciona en forma correcta, úselo para convertir las coordenadas polares $r = 12.5$ y $\theta = 67.8°$ en coordenadas rectangulares.

# Ejercicio 3

Un modelo del crecimiento de la población mundial, en miles de millones de personas, desde 2000 está dado por la ecuación:

$$Población = 6.0\ e^{0.02[año - 2000]}$$

Usando esta fórmula, escriba, compile y ejecute un programa en C++ para estimar la población mundial en el año 2005. Verifique el resultado desplegado por su programa calculando la respuesta en forma manual. Después que haya verificado que su programa funciona en forma correcta, úselo para estimar la población mundial en el año 2012.

# Ejercicio 4

El número de años que se requiere para que se descomponga un cierto isótopo de uranio a la mitad de una cantidad original está dado por la

$$Vida\ media = \ln(2)/k$$

donde $k$ es igual a 0.00012. Usando esta fórmula, escriba, compile y ejecute un programa en C++ que calcule y despliegue la vida media de este isótopo de uranio. Verifique el resultado producido por su programa usando un cálculo manual. Después que haya verificado que su programa funciona en forma correcta, úselo para determinar la vida media de un isótopo de uranio que tenga $k = 0.00026$.

# FUNDAMENTOS DE PROGRAMACIÓN

Fin Clase 03

Profesor: Carlos Díaz