# Deber 4

**Fecha**: 27/05/2019

**Estudiante**: Andrea Garcés

1. Escriba un programa que lea tres valores double distintos de cero, y que determine e imprima si podrían representar los lados de un triángulo.

#include <iostream>

using namespace std;

int Triangulo (double lado\_1, double lado\_2, double lado\_3)

{

if(lado\_1 != 0 && lado\_2 != 0 && lado\_3 != 0)

{

if(lado\_1 < lado\_2 + lado\_3)

{

if(lado\_1 > lado\_2 - lado\_3)

{

cout << "Estos valores pueden formar un triangulo" << endl;

}

else

{

cout << "Estos valores no pueden formar un triangulo" << endl;

}

}

else

{

cout << "Estos valores no pueden formar un triangulo" << endl;

}

}

else

{

cout << "Valores ingresados no validos" << endl;

}

return 0;

}

int main()

{

double lado\_1, lado\_2, lado\_3;

cout << "Ingrese tres valores de mayor a menor: " << endl;

cin >> lado\_1;

cin >> lado\_2;

cin >> lado\_3;

Triangulo(lado\_1, lado\_2, lado\_3);

return 0;

}

1. El factorial de un entero n no negativo se escribe como n! (n factorial) y se define de la siguiente manera: n! = n · (n – 1) · (n – 2) · . . . · 1 ( para valores de n mayores o iguales a 1) y n! = 1 ( para n = 0 o n = 1). Por ejemplo, 5! = 5 · 4 · 3 · 2 · 1, que es 120. Use instrucciones while en cada uno de los siguientes casos:
2. Escriba una aplicación que lea un entero no negativo, que calcule e imprima su factorial.

#include <iostream>

using namespace std;

int Factorial(int x)

{

int result = 1;

int aux = 1;

while(aux <= x)

{

result = result \* aux;

aux++;

}

return result;

}

int main()

{

int num;

int resultado;

cout << "Ingrese un numero:" << endl;

cin >> num;

resultado = Factorial(num);

cout << "El factorial del numero ingresado es: " << resultado << endl;

return 0;

}

b) Escriba un programa que estime el valor de la constante matemática e, utilizando la fórmula:



#include <iostream>

using namespace std;

long double Factorial(int x)

{

int result = 1;

int aux = 1;

while(aux <= x)

{

result = result \* aux;

aux++;

}

return result;

}

long double Obtener\_e(int iterac)

{

long double result = 1;

int aux = 1;

while(aux <= iterac)

{

result = result + (1 / Factorial(aux));

aux++;

}

return result;

}

int main()

{

int n = 15;

long double e;

e = Obtener\_e(n);

cout << "e = " << e << endl;

}

1. Escriba un programa en C++, para realizar cada una de las siguientes tareas:
2. Sumar los enteros impares entre 1 y 99, utilizando una instrucción for. Suponga que se han declarado las variables enteras suma y cuenta.

#include <iostream>

using namespace std;

int Sumar()

{

int suma = 0;

int cuenta;

for(cuenta = 1; cuenta < 100; cuenta += 2)

{

suma = suma + cuenta;

cout << "Cuenta: " << cuenta << endl;

cout << "Suma: " << suma << endl;

}

return suma;

}

int main()

{

cout << "Suma total: " << Sumar() << endl;

return 0;

}

1. Imprimir el valor 333.546372 en una anchura de campo de 15 caracteres, con precisiones de 1, 2 y 3. Imprimir cada número en la misma línea. Justificar a la izquierda cada número en su campo. ¿Cuáles son los tres valores que se imprimen?

#include <iostream>

#include <iomanip>

using namespace std;

int main()

{

long double num = 333.546372;

cout << "El numero es: 333.546372" << endl << endl;

cout << setfill('.');

cout << setw(15) << left << setprecision(6) << num;

cout << "\t\t";

cout << setw(15) << left << setprecision(5) << num;

cout << "\t\t";

cout << setw(15) << left << setprecision(4) << num;

cout << setfill('\0');

cout << endl;

}

1. Calcular el valor de 2.5 elevado a la potencia de 3, utilizando la función pow. Imprimir el resultado con una precisión de 2 en una anchura de campo de 10 posiciones. ¿Que se imprime?

#include <iostream>

#include <iomanip>

#include <math.h>

using namespace std;

int main()

{

cout << fixed << setw(10) << setprecision(2) << setfill('.') << pow(2.5, 3) << endl;

}

1. Imprimir los enteros del 1 al 20, utilizando un ciclo while o for y la variable contador x. Suponga que la variable x se ha declarado, pero no se ha inicializado. Imprimir solamente cinco enteros por línea. [*Sugerencia*: use el cálculo x % 5. Cuando el valor de esta expresión sea 0, imprima un carácter de nueva línea; de lo contrario, imprima un carácter de tabulación.]

#include <iostream>

using namespace std;

int main()

{

int x;

for(x = 1; x <= 20; x++)

{

cout << x << "\t";

if (x % 5 == 0)

{

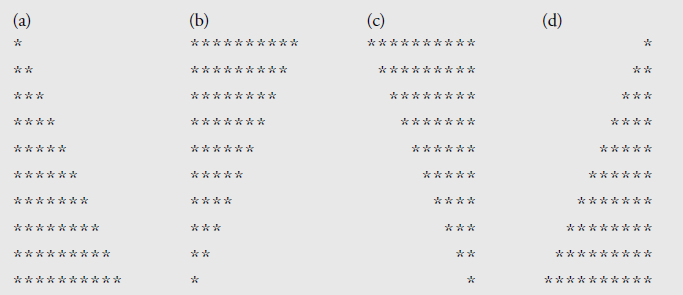
cout << endl;

}

}

}

1. Escriba un programa que utilice ciclos for para generar los siguientes patrones por separado (ver imagen), uno debajo del otro. Use ciclos for para generar los patrones. Todos los asteriscos (\*) deben imprimirse mediante una sola instrucción de la forma cout << '\*'; (esto hace que los asteriscos se impriman uno al lado del otro). [*Sugerencia:* los últimos dos patrones requieren que cada línea empiece con un número apropiado de espacios en blanco. *C*ombine su código de los cuatro problemas separados en un solo programa que imprima los cuatro patrones, uno al lado del otro, haciendo un uso inteligente de los ciclos for anidados.]



(a)

#include<iostream>

using namespace std;

int Triangulo\_a()

{

int filas;

int columnas;

for(filas = 1; filas <= 10; filas++)

{

for(columnas = 1; columnas <= filas; columnas++)

{

cout << "\*";

}

cout << endl;

}

}

int main()

{

Triangulo\_a();

return 0;

}

(b)

#include<iostream>

using namespace std;

int Triangulo\_b()

{

int filas;

int columnas;

for(filas = 1; filas <= 10; filas++)

{

for(columnas = 10; columnas >= filas; columnas--)

{

cout << "\*";

}

cout << endl;

}

}

int main()

{

Triangulo\_b();

return 0;

}

(c)

#include<iostream>

using namespace std;

int Triangulo\_c()

{

int filas;

int columnas;

for(filas = 1; filas <= 10; filas++)

{

for(columnas = 1; columnas <= 10; columnas++)

{

if(columnas >= filas)

{

cout << "\*";

}

else

{

cout << " ";

}

}

cout << endl;

}

}

int main()

{

Triangulo\_c();

return 0;

}

(d)

#include<iostream>

using namespace std;

int Triangulo\_c()

{

int filas;

int columnas;

for(filas = 10; filas >= 1; filas--)

{

for(columnas = 1; columnas <= 10; columnas++)

{

if(columnas >= filas)

{

cout << "\*";

}

else

{

cout << " ";

}

}

cout << endl;

}

}

int main()

{

Triangulo\_c();

return 0;

}

1. Escriba un programa que imprima la siguiente figura de un rombo (ver imagen). Puede utilizar instrucciones de salida que impriman un solo asterisco (\*) o un solo espacio en blanco. Maximice el uso de la repetición (con instrucciones for anidadas), y minimice el número de instrucciones de salida.



#include<iostream>

using namespace std;

int Rombo()

{

int filas;

int columnas;

for(filas = 10; filas >= 1; filas--)

{

for(columnas = 1; columnas <= 10; columnas++)

{

if(columnas >= filas)

{

cout << "\*";

}

else

{

cout << " ";

}

}

for(columnas = 9; columnas >= 1; columnas--)

{

if(columnas >= filas)

{

cout << "\*";

}

else

{

cout << " ";

}

}

cout << endl;

}

for(filas = 1; filas <= 10; filas++)

{

for(columnas = 1; columnas <= 10; columnas++)

{

if(columnas >= filas)

{

cout << "\*";

}

else

{

cout << " ";

}

}

for(columnas = 9; columnas >= 1; columnas--)

{

if(columnas >= filas)

{

cout << "\*";

}

else

{

cout << " ";

}

}

cout << endl;

}

}

int main()

{

Rombo();

return 0;

}

1. *Temperaturas en Centígrados y Fahrenheit:* Implemente las siguientes funciones enteras:
2. El método centígrados que devuelve la equivalencia en grados Centígrados de una temperatura en grados Fahrenheit.

b) La función fahrenheit que devuelve la equivalencia en grados Fahrenheit de una temperatura en grados Centígrados.

c) Utilice estas funciones para escribir un programa que imprima gráficos que muestren los equivalentes en grados Fahrenheit de todas las temperaturas en grados Centígrados, desde 0 hasta 100, y los equivalentes en grados Centígrados de todas las temperaturas en grados Fahrenheit, desde 32 hasta 212. Imprima los resultados en un formato tabular ordenado que minimice el número de líneas de salida, al tiempo que permanezca legible.

*La fórmula de conversión es:*

#include <iostream>

#include <iomanip>

using namespace std;

float Centigrados(float far)

{

float cent;

cent = (far - 32) \* (float) 5/9;

return cent;

}

float Farenheit(float cent)

{

double far;

far = ((float) 9/5 \* cent) + 32;

return far;

}

int Llena\_tabla\_f()

{

float result;

cout << "Tabla conversion: centigrados a farenheit" << endl;

for(int i = 0; i <= 100; i++)

{

result = Farenheit(i);

cout << setw(3) << i << " => ";

cout << setw(5) << result << " | ";

if ((i) % 5 == 0)

{

cout << endl;

}

}

cout << endl;

}

int Llena\_tabla\_c()

{

float result;

cout << "Tabla conversion: farenheit a centigrados" << endl << endl;

for(int i = 32; i <= 212; i++)

{

result = Centigrados(i);

cout << setw(3) << i << " => ";

cout << fixed << setprecision(2) << setw(5) << result << " | ";

if ((i - 1) % 5 == 0)

{

cout << endl;

}

}

}

int main()

{

Llena\_tabla\_f();

cout << endl;

Llena\_tabla\_c();

}

1. Escriba un programa completo en C++ con las dos funciones alternativas que se especifican a continuación, de las cuales cada una simplemente triplica la variable cuenta definida en main. Después compare y contraste ambos métodos. Estas dos funciones son:

a) la función triplicarPorValor, que pasa una copia de cuenta por valor, triplica la copia y devuelve el nuevo valor, y

b) la función triplicarPorReferencia, que pasa cuenta por referencia a través de un parámetro por referencia y triplica el valor original de cuenta a través de su alias (es decir, el parámetro por referencia).

#include <iostream>

#include <iomanip>

using namespace std;

float TriplicarPorValor(float cuenta)

{

cuenta \*= 3;

return cuenta;

}

float TriplicarPorReferencia(float &a)

{

a = a \* 3;

return a;

}

int main()

{

float cuenta = 5;

cout << "Cuenta original: " << cuenta << endl;

cout << "Triplica por valor: " << TriplicarPorValor(cuenta) << endl;

cout << "Cuenta luego de paso por valor: " << cuenta << endl;

cout << "Triplica por referencia: " << TriplicarPorReferencia(cuenta) << endl;

cout << "Valor final de cuenta: " << cuenta << endl;

return 0;

}

1. Escriba un programa en C++ que pida al usuario el radio de un círculo y después llame a la función inline area-Circulo para calcular el área de ese círculo.

#include<iostream>

#include<math.h>

#define PI 3.1415926

using namespace std;

inline float AreaCirculo(float x)

{

float area;

area = (PI \* pow(x, 2));

return area;

}

int main()

{

float radio;

float resultado;

cout << "Ingrese el radio del circulo: " << endl;

cin >> radio;

cout << "El area del circulo es: " << AreaCirculo(radio) << endl;

return 0;

}

1. Escriba un código con un ciclo While que sume los enteros, contando desde 1, y se

detenga cuando la suma sea mayor que 10 000, imprimiendo el entero que se añadió recientemente

a la suma.

#include<iostream>

using namespace std;

int Suma()

{

int num = 1;

int total\_suma = 0;

while(total\_suma <= 10000)

{

total\_suma = total\_suma + num;

num ++;

}

cout << "Ultimo valor agregado: " << (num - 1) << endl;

return total\_suma;

}

int main()

{

int resultado;

resultado = Suma();

cout << "La suma total es: " << resultado << endl;

}

1. Escribir y comprobar un programa que resuelva la ecuación cuadrática (*ax2 + bx + c = 0*).

Tip: Se resuelve con esta fórmula:

#include<iostream>

#include<math.h>

using namespace std;

float Cuadratica(int a, int b, int c)

{

float x1;

float x2;

x1 = ((-b)+(sqrt(pow(b, 2)-(4\*a\*c))))/(2\*a);

x2 = ((-b)-(sqrt(pow(b, 2)-(4\*a\*c))))/(2\*a);

cout << "Las raices de la funcion cuadratica son: " << endl;

cout << "x1 = " << x1 << endl;

cout << "x2 = " << x2 << endl;

return 0.0;

}

int main()

{

int a;

int b;

int c;

cout << "Ingrese tres valores a, b y c, de forma que ax2 + bx + c = 0" << endl;

cout << "a: ";

cin >> a;

cout << "b: ";

cin >> b;

cout << "c: ";

cin >> c;

Cuadratica(a, b, c);

return 0;

}

1. *Escribir un programa que lea los valores de tres lados posibles de un triángulo a, b y c, y calcule en el caso de que formen un triángulo su área y su perímetro, sabiendo que su área viene dada por la siguiente expresión:*

*area* =

*donde p es el semiperímetro del triángulo p* = (*a* + *b* + *c*)/2

#include <iostream>

#include<math.h>

using namespace std;

float Perimetro(float lado\_1, float lado\_2, float lado\_3)

{

float perimetro;

perimetro = lado\_1 + lado\_2 + lado\_3;

return perimetro;

}

float Area(float lado\_1, float lado\_2, float lado\_3)

{

float s\_perimetro;

float area;

s\_perimetro = (lado\_1 + lado\_2 + lado\_3)/2;

area = sqrt((s\_perimetro - lado\_1)\*(s\_perimetro - lado\_2)\*(s\_perimetro - lado\_3));

return area;

}

float Triangulo(float lado\_1, float lado\_2, float lado\_3)

{

if(lado\_1 != 0 && lado\_2 != 0 && lado\_3 != 0)

{

if(lado\_1 < lado\_2 + lado\_3)

{

if(lado\_1 > lado\_2 - lado\_3)

{

cout << "Estos valores pueden formar un triangulo" << endl;

cout << "El perimetro del triangulo formado es: " << Perimetro(lado\_1, lado\_2, lado\_3) << endl;

cout << "El area del triangulo formado es: " << Area(lado\_1, lado\_2, lado\_3) << endl;

}

else

{

cout << "Estos valores no pueden formar un triangulo" << endl;

}

}

else

{

cout << "Estos valores no pueden formar un triangulo" << endl;

}

}

else

{

cout << "Valores ingresados no validos" << endl;

}

return 0.0;

}

int main()

{

float lado\_1, lado\_2, lado\_3;

cout << "Ingrese tres valores de mayor a menor: " << endl;

cin >> lado\_1;

cin >> lado\_2;

cin >> lado\_3;

Triangulo(lado\_1, lado\_2, lado\_3);

return 0;

}

1. *Escribir una función lógica* Vocal *que determine si un carácter es una vocal.*

#include <iostream>

using namespace std;

bool Vocal(char c)

{

bool esVocal;

switch (toupper(c))

{

case 'A':

esVocal = true;

break;

case 'E':

esVocal = true;

break;

case 'I':

esVocal = true;

break;

case 'O':

esVocal = true;

break;

case 'U':

esVocal = true;

break;

default :

esVocal = false;

}

return esVocal;

}

int main()

{

char c;

cout << "Ingrese un caracter: " << endl;

cin >> c;

if (Vocal(c) == true)

{

cout << c << " es una vocal";

}

else

{

cout << c << " no es una vocal";

}

return 0;

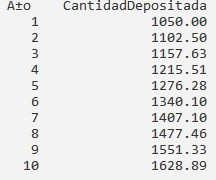
}

1. Una persona invierte $1,000.00 en una cuenta de ahorro que produce 5 por ciento de interés anual. Suponiendo que todo el interés se deposita en la cuenta, cree un programa en c/c++ que calcule e imprima el monto de dinero en la cuenta al final de cada año (tabla), durante los 10 siguientes años. Use la siguiente fórmula para determinar los montos:

Donde:

* + p es el monto que se invirtió originalmente (es decir, el monto principal)
  + t es la tasa de interés anual (por ejemplo, use 0.05 para 5%)
  + n es el número de años
  + c es la cantidad depositada al final del n-ésimo año.

El resultado debe presentarse similar a la siguiente tabla (nota: usar formateadores)



#include <iostream>

#include<math.h>

#include<iomanip>

using namespace std;

float monto(int p, float t, int n)

{

float c;

c = p \* pow((1 + t), n);

return c;

}

int main()

{

int p = 1000, n;

float t = 0.05;

cout << "Año" << "\t\t" << "CantidadDepositada" << endl;

for(n = 1; n <= 10; n++)

{

cout << n << "\t\t" << fixed << setprecision(2) << monto(p, t, n) << endl;

}

return 0;

}