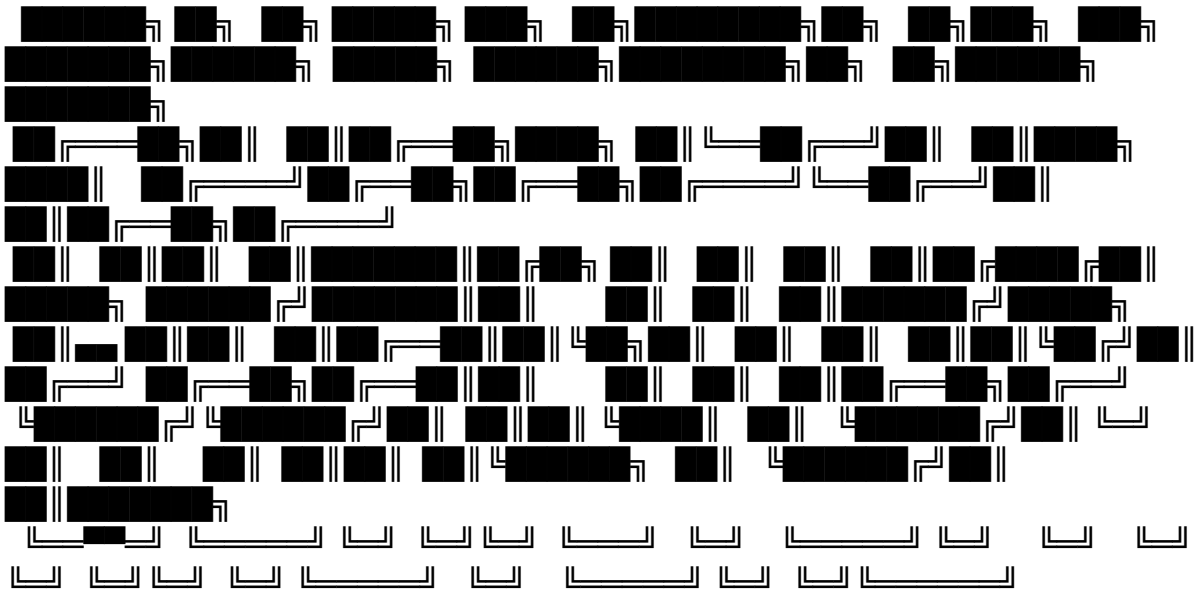
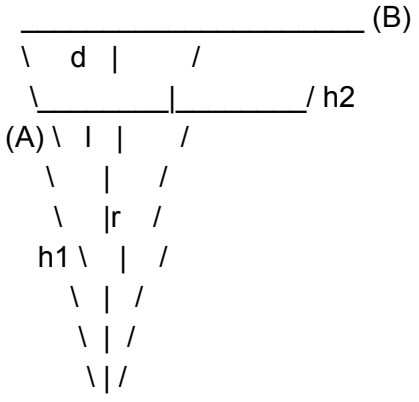
/*



PRESENTA

,
'| |\/| _´_|_ _ _| _ _| _ /\ _ _ ´ _ _ _ _| _ _ /|~) _ |´ _ _ _ _ _\
_|_. | |(/_ | (_)(_|(_) (_|(/_ /~~\| (_||_||| | |(/_(_|(/__\ | |~ (_)||(_|(_)| |(_)_\ |
| \ _| /

Qué pasa chavales, muy buenas a todos, aquí Crespo comentando (xD willy pls). Con este código, escrito en C++, vamos a hacer que
este ordenador calcule pi utilizando el método que nuestro Antiguos usaban: calculando polígonos de lado 2*n (dos veces n).
Si quieres saber las ideas básicas de este algoritmo o no sabes como coj***s has llegado aquí y qué es este código
hablandome así wtf, te recomiendo pasarte por YouTube y verte este vídeo
--> https://youtu.be/DQ5qqHukkAc <--.
Vamos ahora a la miga: Imagina que tenemos DOS polígonos de "2n" lados: Uno inscrito dentro de una circunferencia de radio "r"
(es decir, cuyos vertices tocan la circunferencia, está dentro del circulo) y otro circunscrito (es decir, cuyos lados tocan
la circunferencia, están por fuera del circulo).
En cualquier polígono siempre puedo tirar dos líneas desde el centro a dos vértices contiguos y formar un triángulo
(el menor que se puede formar de esta manera):



al ángulo del triángulo vamos a llamarlo "ang"

centro (O)

...donde:

* r --> radio del circunferencia.
* h2 --> distancia centro-vertice (OB) del poligono que está CIRCUNSCRITO (que es el poligono está por fuera del círculo,
con los lados tocando la circunferencia).
* h1 --> distancia centro-vertice (OA) del polígono que está INSCRITO (que es el poligono está dentro del círculo,
con los vértices tocando la circunferencia).
* l --> es la MITAD del lado del poligono interno.
* d --> es la MITAD del lado del poligono externo.
* ang --> angulo formado por OA y OB. Si el poligono tiene n lados, entonces
sabemos cuanto vale: una vuelta completa son 2*pi, luego si tiene n lados,

$$ang = 2*pi/n$$

Si no ves un carajo del gráfico te recomiendo que estires el tamaño de tu ventana, te llevarás una sorpresa --->
También es bastante recomendable que, si te quiere enterar bien de esto, te lo dibujes en un papel. Really.

Dos definciones más: sea...

* A_n --> el perímetro del polígono de n lados inscrito.
* B_n --> el perímetro del polígono de n lados cricunscrito.

A las cuentas: vamos a calcular cuanto vale "l" y cuanto vale "d". Los necesitaremos para saber cuanto son los perímetros
que es lo que buscamos, ohú. Por trigonometría básica (definición de seno), "l" verifica:

$$l = r * \sin(pi/n)$$

... y "d" verifica:

$$d = r * \tan(\pi/n)$$

Por lo tanto los respectivos perímetros serán el número de lados, n, por el lado, 2l y 2d respectivamente:

$$A_n = 2 * n * r * \sin(\pi/n) \qquad B_n = 2 * n * r * \tan(\pi/n) = A_n / \cos(\pi/n)$$

Entonces, el perímetro de un polígono con el doble de lados debe verficar:

$$A_{2n} = 4 * n * r * \sin(\pi/2n) \qquad B_{2n} = 4 * n * r * \tan(\pi/2n) = A_{2n} / \cos(\pi/2n)$$

El objetivo, ahora, es dejar A_2n y B_2n en función de A_n y B_n, es decir, que no dependamos de calcular ángulos (que
requieren de pi) para saber cuanto vale los perímetros. De este modo partiendo de un polígono muy sencillo cuyas dimensiones puedan
ser cálculados usando identidades trigonometricas muy sencillas, podemos ir generando los polígonos de más y más lados.

No voy a escribir explicitamente como poner unos en función de los otros; es un desarrollo largo y tengo vídeos que hacer! Pero
si queréis intentarlo, usad todas las identidades trigonométricas que podáis, especialmente las del angulo doble y ángulo mitad ;)

El resultado es:

$$B_{2n} = 2 * A_n * B_n / (A_n + B_n)$$
$$A_{2n} = \text{raiz cuadrada de } \{ A_n * B_{2n} \}$$

Dado que A se acerca al perimetro de la circunferencia desde valores menores y B desde valores mayores, podemos usar estos para
calcular los errores del método. Es decir, que, una vez paremos en un cierto número de lados, tomaremos la media de ambos pi's
calculados con los diferentes perímetros como el valor de este, y la diferencia de estos como el error que estamos comentiendo;
así tendremos el rango de valores en lo que el verdadero pi se encuentra.

Quiero agradecer ENORMEMENTE a Diego Soler Polo por ayudarme a encontrar este algoritmo. Sin él, muchos aspectos de este vídeo
no serían posibles... y este código menos.

*/

```
#include <iostream>  // Cargo librerías (colecciones de funciones ya hechas que hacen
cositas): "iostream" me permite sacar texto
#include <cmath>     // y números por el terminal, para que puedas ver el valor de pi, y
"cmath" es una colección de funciones
#include <fstream>   // matemáticas que necesito, como elevar al cuadrado y hacer la raiz
cuadrada.

using namespace std;  // A efectos práticos, esto es para no tener que poner "std" todo el
rato.

int main(){          // ¡Comenzamos!

   // Imprimimos en el terminal... la bienvenida ;)

   cout<<endl<<endl;
   cout<<"                ╔═╗ ┬ ┬┌─┐ ┌┐ ┌┌┬┐┬ ┬┌┬┐  ╔═╗┬─┐ ┌─┐ ┌─┐ ┌┬┐┬
┬┬─┐ ┌─┐ "<<endl;
   cout<<"                ║═╗├─┤ │ │├─┤│││ │ │││││ ╠╣ ├┬┘├─┤ │ │
│ ├┬┘ ├┤ "<<endl;
   cout<<"                ╚═╝╚═╝└─┘┴ ┴┴ ┴└┘ ┴ └─┘┴ ┴ ╚ ┴└─┴─┴ ┴└─┘ ┴
 └─┘┴ └─ └─┘ "<<endl;
   cout<<"
______________________________________________________________________
"<<endl<<endl;
   cout<<"                              PRESENTA...                         "<<endl;
   cout<<endl;
   cout<<"          ╔═╗┌─┬ ┌─┬┬┬ ┌─┐  ┌┬┐┌─┐ ╔═╗┬    ╔═╗┌─┬
┬┌─┐ ┌─┐ ┌┐ ┌┌─┐ ┌─┐"<<endl;
   cout<<"          ║ ├─┤│ │ │││ ││ │││├┤  ╠═╝│ ── ╠═╝│ ││ ││┬│ │││││
│ └─┐"<<endl;
   cout<<"          ╚═╝┴ ┴┴─┘ └─┘ └─┘┴─┘ └─┘ ─┴┘└─┘ ╩ ┴    ╩
 └─┘┴─┘┴└─┘ └─┘┘ ┴ └─┘ └─┘"<<endl;

   cout<<endl<<"                   Dividiendo perímetros entre diametros desde
2017"<<endl<<endl;


cout<<"_________________________________________________________________
____________________________"<<endl<<endl;
   cout<<"                                ADVERTENCIA                         "<<endl;
   cout<<"Si está usando este código en un compilador online, puede ser que utilizar un
número excesivo"<<endl;
   cout<<" de lados haga que el cálculo necesite tanto tiempo para ser computado que el
compilador lo"<<endl;
   cout<<"             aborte automáticamente. En ese caso, pruebe un número
menor."<<endl;
```

```
   ////////////////////////////////////////////////////////////////////////////////
   ////////////////////////////////////////////////////////////////////////////////
   ////////////////////////////////////////////////////////////////////////////////

   int n = 12345  ;        // NÚMERO DE LADOS DEL POLÍGONO. ¡CAMBIAD ESTO!

   ////////////////////////////////////////////////////////////////////////////////
   ////////////////////////////////////////////////////////////////////////////////
   ////////////////////////////////////////////////////////////////////////////////

   // Se expulsa tal número por terminal:

   cout<<endl<<"
╔═══════════════════════════════════════════════════════════════════════════════
═══════════════════════════════════════════════╗";
   cout<<endl<<"  ║              Número de lados máximo del polígono : "<<n;
   cout<<endl<<"
╚═══════════════════════════════════════════════════════════════════════════════
═══════════════════════════════════════════════╝";

   double r=1;  // Este es RADIO DE LA CIRCUNFERENCIA. Puedes cambiarlo; el valor de
pi no se ve afectado.

   double A;    // Defino los Perímetros de los polígonos inscritos y circunscritos.
   double B;

   A=4*sqrt(2)*r;  // Cargo los perímetros de un CUADRADO inscrito (A) y circunscrito (B).
Estos valores pueden ser fácilmente
   B=8*r;          // obtenidos con el teorema de pitagoras, no pi required.

   double m=4;     // Este es el número de lados de los polígonos con los que estamos
trabajando.

   while (m*2<=n) {  // BUCLE. Si el número de lados del polígono a generar supera el
número impuesto por usuario, para.

      B=2*A*B/(A+B);      // Calculo de los perímetros con el doble de lados. A cada vuelta los
valores de A y B se sobreeescriben.
      A=sqrt(A*B);

      m=m*2;              // El número de lados se duplica en cada vuelta.

   }              // Fin de BUCLE
```

```
   double pi=(  A/2/r + B/2/r  )/2;         // Calculamos pi como la media de los valores de pi de cada perimetro...
   double err = abs(  A/2/r - B/2/r  )/2;   // ... y el error como la resta.

   cout.precision(15); // Estos son el número de digitos que quiero que se expulsen por pantalla. Puedes aumentarlo si quieres.

   // Sacamos los resultados por pantalla para disfrute del usuario:

   cout<<endl<<"
╔═══════════════════════════════════════════════════════════════════════════════════════════════════════════════════════╗";
   cout<<endl<<"  ║              Con un polígono de "<<m<<" lados, obtenemos:";
   cout<<endl<<"  ║";
   cout<<endl<<"  ║                    "<<"Pi = "<<pi<<"  +/-  "<<err;
   cout<<endl<<"  ║";
   cout<<endl<<"  ║              "<<"o, dicho de otra manera, el valor de pi se encuentra entre";
   cout<<endl<<"  ║                    "<<pi+err<<"   y   "<<pi-err;
   cout<<endl<<"
╚═══════════════════════════════════════════════════════════════════════════════════════════════════════════════════════╝"<<endl<<endl;


   return 0;  // Y hemos terminado. Cerramos el chiringuito.

}
```