```
//sumar las cifras de un entero
#include<iostream>
using namespace std;
int main()
{
  //notar la inicializacion de suma
  int n,cifra,suma=0;
  cout<<"Introduce un numero entero:";
  cin>>n;
  //mientras quedan cifras
  while(n!=0)
    {
      //obtener la cifra mas a la derecha
      cifra=n%10;
      //sumarla
      suma=suma+cifra;
      //eliminar la cifra mas a la derecha
      n=n/10;
    }
  cout<<"Suma de las cifras:"<<suma<<endl;
}
```

```
//invertir el orden de las cifras
//de un numero entero
#include<iostream>
using namespace std;
int main()
{
  //notar la inicializacion de reves
  int n,cifra,reves=0;
  cout<<"Introduce un numero entero:";
  cin>>n;
  //mientras quedan cifras
  while(n!=0)
    {
      //obtener la cifra mas a la derecha
      cifra=n%10;
      //sumarla
      reves=reves*10+cifra;
      //eliminar la cifra mas a la derecha
      n=n/10;
    }
  cout<<"Suma de las cifras:"<<reves<<endl;
}
```

```
//averiguar si un numero es o no capicua
#include<iostream>
using namespace std;
int main()
{
  //notar la inicializacion de reves
  int n,copia,cifra,reves=0;
  cout<<"Introduce un numero entero:";
  cin>>n;
  //se hace una copia del numero
  //el proceso que lo vuelve del reves
  //es destructivo
  copia=n;
  //mientras quedan cifras
  while(copia!=0)
    {
      //obtener la cifra mas a la derecha
      cifra=copia%10;
      //sumarla
      reves=reves*10+cifra;
      //eliminar la cifra mas a la derecha
      copia=copia/10;
    }
  //si el numero del reves es igual al original
  //es capicua
  if (reves==n)
    cout<<"Es capicua"<<endl;
  else
    cout<<"No es capicua"<<endl;
}
```

```
#include<iostream>
using namespace std;
//convierte decimal en binario usando
//ndigitos como maximo
//el resultado se codifica como entero
//no comprueba si existe desbordamiento
int main()
{
  int a,copia,ndigitos;
  cout<<"introduce un decimal:";
  cin>>a;
  copia=a;
  cout<<"numero de digitos de la representacion binaria:";
  cin>>ndigitos;
  int b=0,peso=1;
  //peso del digito binario mas significativo
  for (int i=1;i<ndigitos;i++)
    peso*=2;
  //el valor de los digitos se restan de a
  while(peso!=0)
    {
      //si el peso es menor o igual
      //el digito binario es 1
      if (peso<=a)
        {
          //se resta el peso
          a=a-peso;
          //se introduce un uno a la derecha
          b=b*10+1;
        }
      else
        //en caso contrario  se introduce un 0
        b=b*10;
      //peso inmediatamente inferior
      peso/=2;
    }
  cout<<"El decimal:"<<copia<<" en binario con "
      <<ndigitos<<" digitos es:"<<b;
}
```

```
//Obtener los 20 primeros términos de
//la sucesión de Fibonacci.
#include<iostream>
using namespace std;
int main()
{
  unsigned long int fn,fn_1=1,fn_2=1;
  //los dos primeros son por definicion
  cout<<"F(0)="<<1<<endl<<"F(1)="<<1<<endl;
  //los siguientes 18
  for (int i=2;i<=19;i++)
    {
      //siguiente
      fn=fn_1+fn_2;
      cout<<"F("<<i<<")="<<fn<<endl;
      //actualizacion de los anteriores
      fn_2=fn_1;
      fn_1=fn;
    }
}
```

```
//Aproximación de e^x por una serie,
//finaliza cuando se alcanza un máximo
//número de iteraciones o una determinada
//precisión, lo que suceda antes.
//Intentar realizar el mínimo número de
//cálculos posible.
#include<iostream>
using namespace std;
int main()
{
  //variables para el exponente y precision
  float x, precision;
  //numero de iteraciones
  int iteraciones;
  cout<<"Introduce x, precision, iteraciones maximas:";
  cin>>x>>precision>>iteraciones;
  //termino actual, anterior,
  //resultado, indice del termino
  float t_n=1,t_n_1,resultado=0,n=0;
  while(t_n>precision&&n<iteraciones)
    {
      resultado+=t_n;
      t_n_1=t_n;
      n++;
      t_n=t_n_1*x/n;
    }

  cout<<"e elevado a "<<x<<" con precision "<<precision<<
    " iteraciones="<<n<<'='<<resultado;
}
```

```cpp
//mostrar los n primeros capicuas
//n se pide por el teclado
#include<iostream>
using namespace std;
int main()
{
  //notar la inicializacion de n, comenzamos a probar
  //por el primero, el cero
  int n=0,copia,cifra,reves,contador=0,cantidad;
  cout<<"Cuantos capicuas quieres:";
  cin>>cantidad;
  //mientras no encuentre 'cantidad' capicuas
  while(contador<cantidad)
    {
      //ver capicua.cpp
      copia=n;
      reves=0;
      //mientras quedan cifras
      while(copia!=0)
        {
          //obtener la cifra mas a la derecha
          cifra=copia%10;
          //sumarla
          reves=reves*10+cifra;
          //eliminar la cifra mas a la derecha
          copia=copia/10;
        }
      //si el numero del reves es igual al original
      //es capicua
      if (reves==n)
        {
          //lo cuento
          contador++;
          //lo muestro
          cout<<n<<endl;
        }
      //siguiente numero para probar
      n++;
    }
}
```

```cpp
//Determinar los numeros menores que 1000 tales que
//aparecen al final de su cuadrado
#include<iostream>
using namespace std;
//numeros<1000 tales que aparecen
//al final de su cuadrado
int main()
{
  //analizar los numeros menores que 1000
  for (int i=0;i<1000;i++)
    {
      //elevar el numero al cuadrado
      int cuadrado=i*i,copia=i;
      //inicialmente suponemos que aparece al final
      bool aparece_al_final=true;
      //comparar cifras empezando por el final
      //si se encuentra una distinta
      //no hace falta seguir
      while (copia!=0&&aparece_al_final)
        {
          //si alguna es distinta,
          //no aparece al final
          if (copia%10!=cuadrado%10)
            aparece_al_final=false;
          //eliminamos las cifras mas a la derecha
          copia/=10;
          cuadrado/=10;
        }
      if (aparece_al_final)
        cout<<i<<endl;
    }
}
```

```cpp
//Determinar cual es la fraccion que se aproxima
//mas a PI con el numerador y el denominador
//menores que 500
#include<iostream>
using namespace std;
#include<cmath>
//valor de PI
const float PI=acos(-1.0);
int main()
{
  //dmin puede inicializarse a cualquier dif
  //que sepamos se va a producir
  float dmin=PI-1;
  int numPI,denPI,i,j;
  //diferentes combinaciones de
  //denominador y denominador
  for (i=1;i<500;i++)
    for (j=1;j<i/2;j++)
      {
        //si se encuentra un cociente
        //que se aproxime mas a PI
        if (abs((float)i/j-PI)<dmin)
          {
            //guardo el numerador y el denominador
            numPI=i;
            denPI=j;
            //calculo la nueva distancia mínima
            dmin=abs((float)i/j-PI);
          }
      }
  cout<<PI<<"~="<<(float)numPI/denPI<<"="
        <<numPI<<'/'<<denPI<<endl;
}
```

```
//Descomponer un número en factores primos
#include<iostream>
#include<cmath>
using namespace std;
int main()
{
  int n,divisor=2;
  cout<<"Introduce un numero entero:";
  cin>>n;
  //mientras el divisor sea menor o igual que el numero
  while(divisor<=n)
    {
      int potencia=0;
      //mientras el numero sea divisible
      //entre el divisor
      while(n%divisor==0)
        {
          //se divide
          n/=divisor;
          //se cuenta el numero de veces
          potencia++;
        }
      //si se ha contado al menos una vez
      //es un factor primo
      if (potencia>0)
        cout<<divisor<<'^'<<potencia<<endl;
      divisor++;
    }
}
```

```
//Un número perfecto es aquel que
//coincide con la suma de todos sus divisores,
//incluyendo el 1 pero no el propio número
//(por ejemplo 6 = 1+2+3, 28 = 1+2+4+7+14).
//Escribir un programa que, muestre por la
//pantalla los números perfectos menores que 100.
#include<iostream>
using namespace std;
int main()
{
  //para cada numero entre 1 y 99
  for (int i=1;i<100;i++)
    {
      //inicializamos la suma a 1
      int suma=1;
      //para cada candidato a divisor
      //excepto el mismo
      for (int j=2;j<=i/2;j++)
        //si es divisor
        if (i%j==0)
          //Se suma
          suma+=j;
      //si la suma coincide con el numero
      //cumple la condicion
      if (suma==i)
        cout<<i<<endl;
    }
}
```

```
//Buscar el mayor número entero
//(menor que 45000 por cuestiones de rango)
// cuyo cuadrado se escriba con
//dígitos distintos.
//Ejemplos de cuadrados que cumplen esa condición:
//13^2 = 169 286^2=81796 32043^2=1026753849
#include<iostream>
using namespace std;
int main()
{
  //esta variable se usa para salir del
  //bucle cuando se encuentra el primero
  bool encontrado=false;
  for (int i=44999;i>0&&!encontrado;i--)
    {
      int cuadrado=i*i;
      //inicialmente suponemos que se
      //cumple la condicion
      bool digitos_distintos=true;
      //mientras queden digitos en el numero
      while(cuadrado!=0)
        {
          //cifra menos significativa de cuadrado
          //y el resto de las cifras
          int cifra=cuadrado%10,
              resto_cifras=cuadrado/10;
          //mientras queden cifras en resto_cifras
          while(resto_cifras!=0)
            {
              //si coincide alguna con la
              //menos significativa de cuadrado
              if (cifra==resto_cifras%10)
                //ya no son todos distintos
                digitos_distintos=false;
              //siguiente cifra de resto_cifras
              resto_cifras/=10;
            }
          //siguiente cifra de cuadrado
          cuadrado/=10;
        }
      //si digitos_distintos sigue siendo true
      if (digitos_distintos)
        {
          //lo hemos encontrado
          encontrado=true;
          //lo mostramos
          cout<<i<<"^2="<<i*i<<endl;
        }
    }
}
```