

Practica 1

Esta practica consta de 5 ejercicios. Aquí escribiré los códigos fuente (en azul) y comentare los resultados obtenidos al ejecutarlos(en negro). Los archivos *.cpp y los archivos *.exe están en el mini cd adjunto.

Ejercicio 1(e_1.cpp/e_1.exe);

```
/*(Practica 1)e_1.cpp.
*Programa realizado por Marc Belda.
*/
#include <iostream>
#include <cmath>
#include <fstream>
using namespace std;
int main()
{
float a, b, y, xmax, xmin;
float ppintervalo, paso;
ofstream fout ("recta_e.dat");
cout <<"Este programa obtiene pares de puntos para una funcion\n";
cout <<"f(x)=a*x+b, donde a es la pendiente de la recta y b la ordenada\n";
cout <<"en el origen. Elegiremos los valores de a y b, los extremos\n";
cout <<"de la recta(xmax y xmin), y el numero de puntos que calcularemos\n";
cout <<"en ese intervalo. Elija el valor de los parametros a y b.\n";
cout <<"Valor de la pendiente a, a=\n";
cin >> a;
cout <<"Valor de la ordenada en el origen b, b=\n";
cin >> b;
cout <<"Elija los extremos del intervalo, xmax y xmin.\n";
cout <<"Valor de xmax, xmax=\n";
cin >> xmax;
cout <<"Valor de xmin, xmin=\n";
cin >> xmin;
cout <<"Elija ahora el n° de puntos dentro de ese intervalo que calcularemos\n";
cout <<"para representar la recta, ppintervalo=\n";
cin >> ppintervalo;
paso = (xmax - xmin)/ ppintervalo;
float x = xmin;
while (x <= xmax)
{
y = (a*x) + b;
cout <<x<<"t"<<y<<"\n";
fout <<x<<"t"<<y<<"\n";
x = x + paso;
}
cout <<"Los datos han sido guardados en el archivo recta_e.dat.\n";
return 0;
```

}

Comprobamos el funcionamiento del programa eligiendo distintos valores de la pendiente (a) y de la ordenada en el origen (b).

Ejemplo 1;

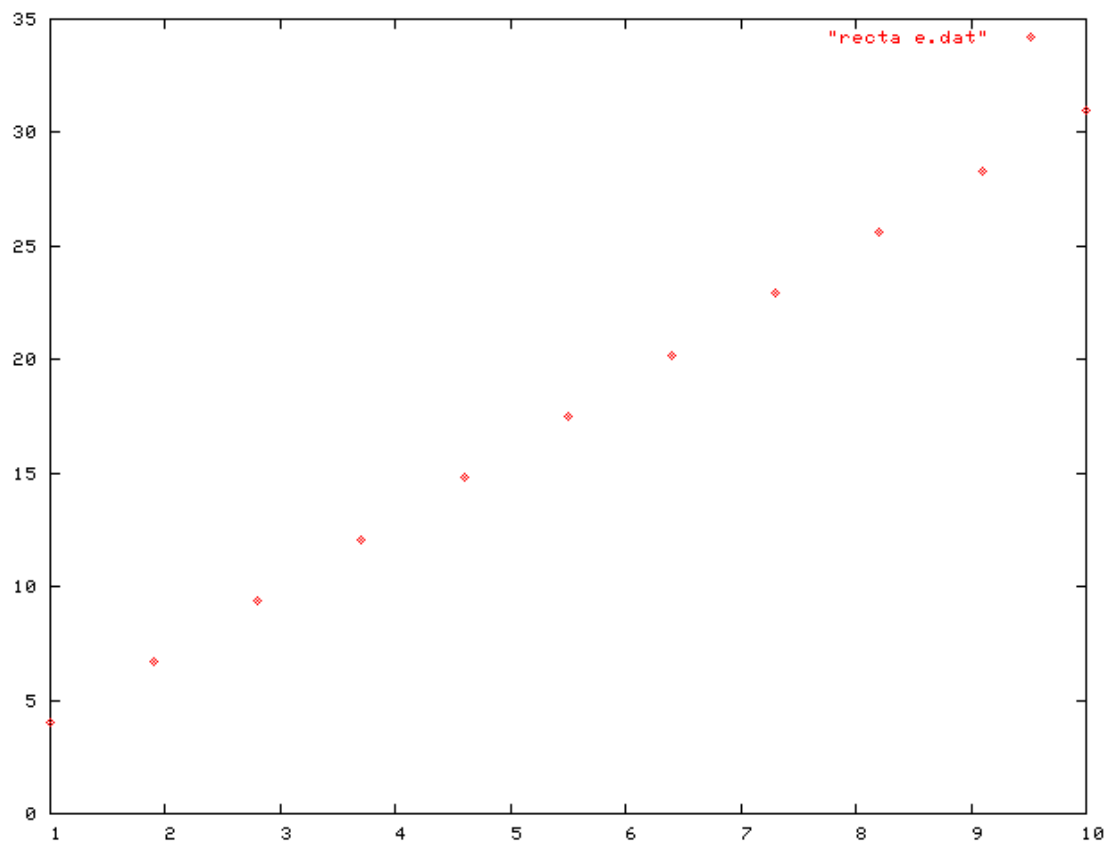
$$a = 3$$

$$b = 1$$

El programa escribe la siguiente tabla $x, f(x)$;

1	4
1.9	6.7
2.8	9.4
3.7	12.1
4.6	14.8
5.5	17.5
6.4	20.2
7.3	22.9
8.2	25.6
9.1	28.3
10	31

Y el Gnuplot dibuja la siguiente grafica;



Ejemplo 2;

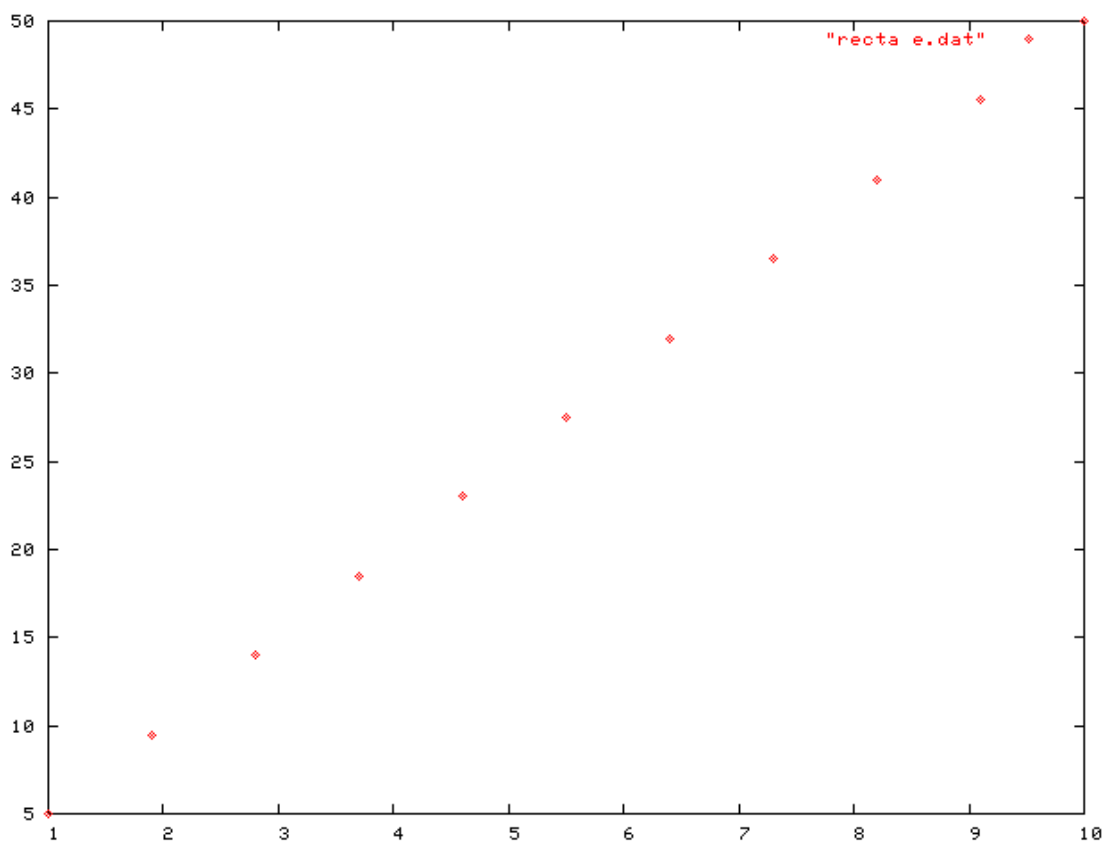
$a = 5$

$b = 0$

El programa escribe la siguiente tabla $x, f(x)$;

1	5
1.9	9.5
2.8	14
3.7	18.5
4.6	23
5.5	27.5
6.4	32
7.3	36.5
8.2	41
9.1	45.5
10	50

Y el Gnuplot dibuja la siguiente grafica;



Ejercicio 2(e_2.cpp/e_2.exe);

```
/*(Practica 1)e_2.cpp
*Programa realizado por Marc Belda.
*/
#include <iostream>
#include <cmath>
using namespace std;
int main ()
{
float a, b, c;
float x1, x2;
cout <<"Este programa calcula las raices reales(x1 y x2)de un polinimio de segundo
grado\n";
cout <<"a*x**2+b*x+c=0.Elegiremos los valores de los parametros a,b y c.\n";
cout <<"Elija el valor de a, a=\n";
cin >> a;
cout <<"Elija el valor de b, b=\n";
cin >> b;
cout <<"Elija el valor de c, c=\n";
cin >> c;
float k = ((b*b)-(4*a*c));
if (k >= 0)
{
x1 = (-b+sqrt(k))/(2*a);
x2 = (-b-sqrt(k))/(2*a);
cout <<"El polinomio selecionado tiene soluciones reales x1,x2.\n";
cout <<"x1=\n";
cout <<x1<<"\n";
cout <<"x2=\n";
cout <<x2<<"\n";
}
else if (k < 0)
{
cout <<"El valor del discriminante(k)es menor que cero.\n";
cout <<"k=\n";
cout <<k<<"\n";
cout <<"El polinomio tiene raices complejas.Este programa no sabe calcularlas.\n";
}
return 0;
}
```

Comprobamos el funcionamiento del programa para dos ecuaciones de segundo grado. Una de ellas con las dos soluciones reales y la otra con dos soluciones complejas.

Ejemplo 1;

En el primer caso evaluaremos la ecuación;

$$x^2 - x - 12 = 0$$

Esta ecuación de segundo grado tiene dos soluciones reales;

$$x_1 = 4$$
$$x_2 = -3$$

Ahora evaluaremos la ecuación;

$$x^2 + 2x + 2 = 0$$

Esta ecuación de segundo grado tiene dos soluciones complejas;

$$x_1 = -1 + i$$
$$x_2 = -1 - i$$

Estos resultados son correctos.

Ejercicio 3(e_3.cpp/e_3.exe);

```
/*(Practica 1)e_3.cpp
*Programa realizado por Marc Belda
*/
#include <iostream>
#include <cmath>
#include <fstream>
using namespace std;
int main()
{
float a, b, c, y, xmax, xmin;
float ppintervalo, paso;
ofstream fout ("curva_e.dat");
cout << "Este programa obtiene pares de puntos par un polinomio de\n";
cout << "segundo grado(f(x)=a*x**2+b*x+c).\n";
cout << "Elegiremos los parametros a,b,c,los extremos de la curva\n";
cout << "(xmax y xmin) y l nº de puntos que calcularemos en ese intervalo.\n";
cout << "Elija el valor de los parametros a,b y c.\n";
cout << "Valor de a, a=\n";
cin >> a;
cout << "Valor de b, b=\n";
cin >> b;
cout << "Valor de c, c=\n";
cin >> c;
cout << "Elija los extremos del intervalo,xmax y xmin.\n";
cout << "Valor de xmax, xmax=\n";
cin >> xmax;
```

```

cout <<"Valor de xmin, xmin=\n";
cin >> xmin;
cout <<"Elija ahora el nº de puntos dentro de ese intervalo que calcularemos\n";
cout <<"para representar la curva,ppintervalo=\n";
cin >> ppintervalo;
paso = (xmax - xmin)/ppintervalo;
float x = xmin;
while (x <= xmax)
{
y = (a*x*x) + b*x +c;
cout <<x<<"\t"<<y<<"\n";
fout <<x<<"\t"<<y<<"\n";
x = x + paso;
}
cout <<"Los datos han sido guardados en el archivo curva_e.dat.\n";
return 0;
}

```

Comprobaremos el funcionamiento de este programa dibujando con el Gnuplot las parábolas del ejercicio anterior.

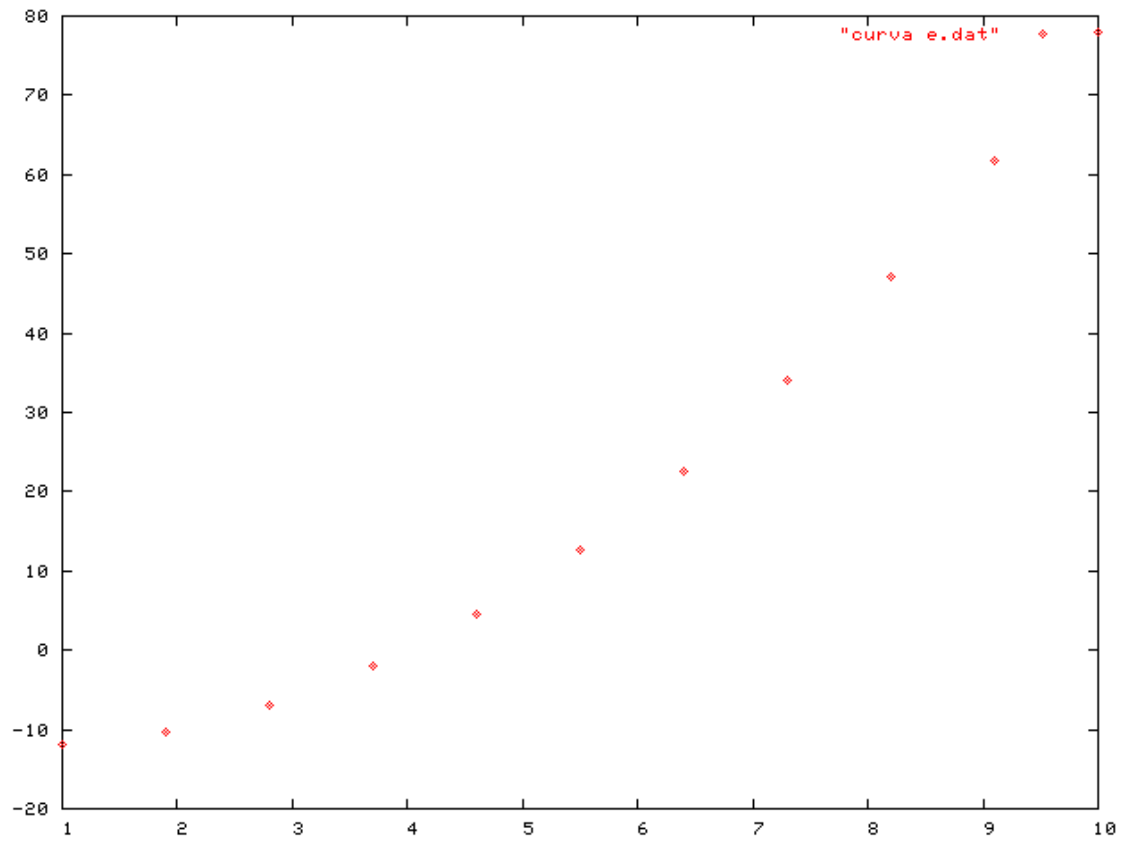
Ejemplo 1;

$$x^2 - x - 12 = f(x)$$

El programa escribe la siguiente tabla $x, f(x)$;

1	-12
1.9	-10.29
2.8	-6.96
3.7	-2.01
4.6	4.56
5.5	12.75
6.4	22.56
7.3	33.99
8.2	47.04
9.1	61.71
10	78

Y el Gnuplot dibuja la siguiente grafica;



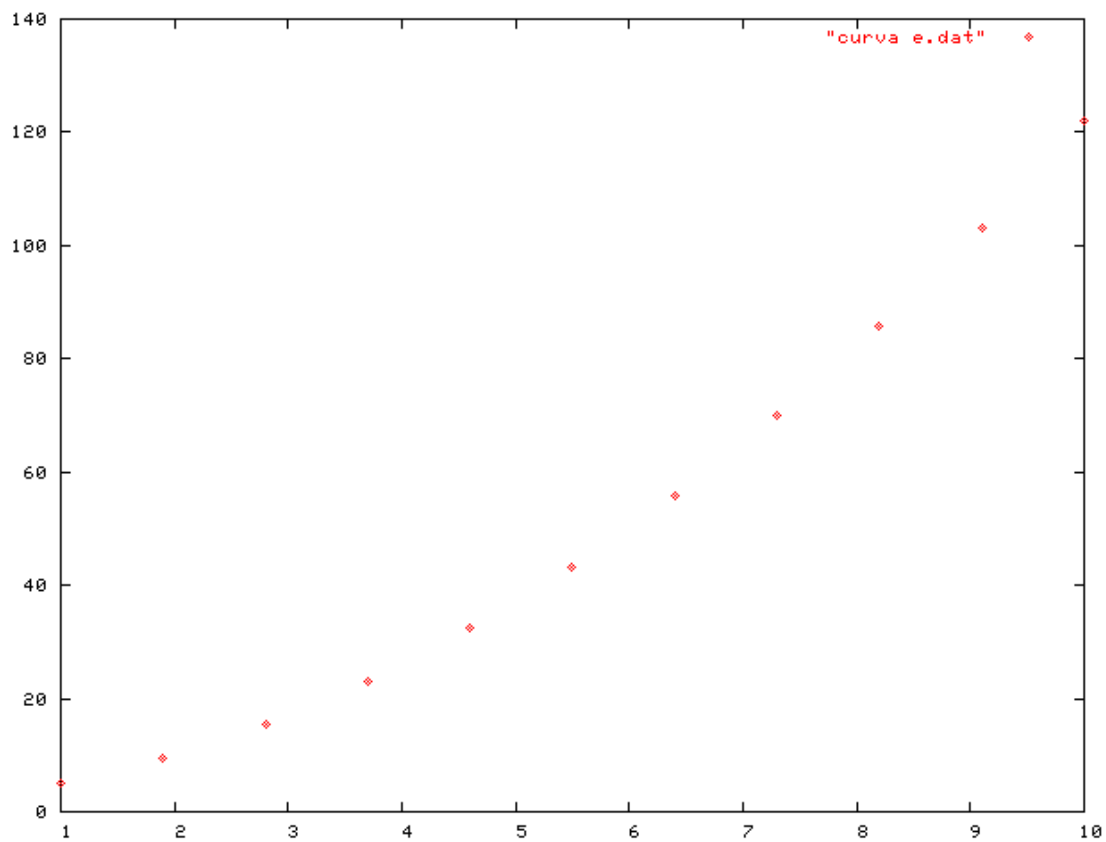
Ejemplo 2;

$$x^2 + 2x + 2 = f(x)$$

El programa escribe la siguiente tabla $x, f(x)$;

1	5
1.9	9.41
2.8	15.44
3.7	23.09
4.6	32.36
5.5	43.25
6.4	55.76
7.3	69.89
8.2	85.64
9.1	103.01
10	122

Y el Gnuplot dibuja la siguiente grafica;



Ejercicio 4(e_4.cpp/e_4.exe);

```
/*(Practica 1)e_4.cpp.  
*Programa realizado por Marc Belda.  
*/  
#include <iostream>  
#include <cmath>  
#include <string>  
using namespace std;  
int main()  
{  
    string respuesta;  
    const double tolerancia= 1.0e-6;  
    double y;  
    int i;  
    cout <<"Este programa calcula la raiz cubica de un n°\n";  
    do
```



```

{
cout << "Introduce el nº cuya raíz cubica quieres calcular, y=\n";
cin >> y;
double x = y;
double z = pow(x, 1/3.0);
i=1;
int iteraciones = 0;
do
{
x = ((2*x) + (y/pow(x,2))) / 3 ;
iteraciones = iteraciones + i;
}
while (abs(y - pow(x,3)) > tolerancia);
cout << "La raíz cubica exacta de " << y << " es " << z << endl;
cout << "La misma raíz cubica calculada con el metodo de Newton-Rapshow es," << x << endl;
cout << "Teniendo en cuenta que la tolerancia de calculo ha sido," << tolerancia << endl;
cout << "El nº de iteraciones utilizado en el calculo aproximado ha sido," << iteraciones << endl;
cout << "¿Desea calcular la raíz cubica de otro nº utilizando el proceso anterior?\n";
cout << "Si la respuesta es si, presione la tecla s o S. En caso contrario presione cualquier otra tecla.\n";
cin >> respuesta;
}
while (respuesta=="s" || respuesta=="S");
return 0;
}

```

Comprobamos el funcionamiento del programa calculando la raíz cúbica de tres números.

La raíz cúbica exacta de 27 es 3, el valor hallado mediante nuestro algoritmo es 3 con un total de 9 iteraciones.

La raíz cúbica exacta de 59.63 es 3.9068, el valor hallado mediante nuestro algoritmo es 3.9068 con un total de 11 iteraciones.

La raíz cúbica exacta de 1005.12 es 10.017, el valor hallado mediante nuestro algoritmo es 10.017 con un total de 15 iteraciones.

Como vemos, con un número de iteraciones adecuado, el método es capaz de calcular la raíz cúbica exacta.

Ejercicio 5(e_5.cpp/e_5.exe);

```

/*(Practica 1)e_5.cpp.
*Programa realizado por Marc Belda.
*/
#include <iostream>
#include <cmath>
#include <string>
using namespace std;

```

```

int main()
{
string respuesta;
double tolerancia;
double y;
int i , iteracmax;
cout <<"Este programa calcula la raiz cubica de un n°\n";
do
{
cout <<"Introduce el n°cuya raiz cubica quieres calcular: ";
cin >> y;
cout <<"Introduce la tolerancia de calculo que deseas aplicar: ";
cin >> tolerancia ;
cout <<"Introduce el n° max de iteraciones: ";
cin >> iteracmax;
double x = y;
double z = pow(x,1/3.0);
i=1;
int iteraciones = 0;
do
{
x = ((2*x)+ (y/pow(x,2))) / 3 ;
iteraciones = iteraciones + i;
}
while (abs(y - pow(x,3)) > tolerancia && iteraciones <iteracmax);
cout <<"La raiz cubica exacta de " << y << " es " << z << endl;
cout <<"La misma raiz cubica calculada con el metodo de Newton-Rapshow es," << x <<
endl;
cout <<"Teniendo en cuenta que la tolerancia de calculo ha sido," << tolerancia << endl;
cout <<"El n° de iteraciones utilizado en el calculo aproximado ha sido," << iteraciones <<
endl;
cout <<"¿Desea calcular la raiz cubica de otro n°utilizando el proceso anterior?\n";
cout <<"Si la respuesta es si,presione la tecla s o S.En caso contrario presione cualquier
otra tecla.\n";
cin >> respuesta;
}
while (respuesta=="s" ||respuesta=="S");
return 0;
}

```

Este programa es una variante del anterior donde, además, podemos elegir la precisión del calculo y el numero máximo de iteraciones del algoritmo.

A modo de ejemplo he calculado la raíz cúbica de 27 con la misma precisión de calculo que en ejercicio anterior pero utilizando solo 4 iteraciones. El resultado es que la raíz cúbica de 27 es igual a 5.52838. Este resultado demuestra que el número de iteraciones es vital para conseguir la precisión deseada.