UNIVERSIDAD AUTÓNOMA DEL ESTADO DE MÉXICO

FACULTAD DE INGENIRÍA

Ingeniería en Computación

“Métodos Numéricos”

Trabajo:

Programa 1: Programa para solucionar un caso de ecuaciones no lineales

Aplicación Método de la Secante

DOCENTE:

María de los Ángeles Contreras Flores.

ALUMNO:

Benigno González Emmanuel

Grupo: 11

Ciclo escolar: 2021-A

Fecha: 09 de Marzo de 2021

METODOLOGÍA DE LA PROGRAMACIÓN ORIENTADA A OBJETOS

FASE 1: INGENIERÍA DE REQUISITOS

Planteamiento del Problema

Calcular la masa (m) según lo siguiente;

La velocidad de un paracaidista que cae está dada por:

$$v = \frac{gm}{c}(1 - e^{-(c/m)t})$$

dónde: g = 9.81 m/s.

Para un coeficiente de arrastre de c = 15 kg/s, calcular la masa m de modo que la velocidad sea v = 35 m/s en t = 9s.

Para aproximar el valor de la masa, el usuario deberá ingresar el valor de la velocidad, tiempo, coeficiente de arrastre el número de iteraciones deseadas y el rango en donde se estimará la raíz.

$$f(m) = \left(\frac{g * (m)}{c} * \left(1 - e^{-\left(\frac{c}{m}\right)*t}\right)\right) - v$$

$$f(m) = \left(\frac{9.81 * (m)}{15} * \left(1 - e^{-\left(\frac{15}{m}\right)*9}\right)\right) - 35$$

Propuesta de Solución

Se propone desarrollar un programa que aproxime el valor de la masa de un paracaidista que cae, aplicando el **Método de la Secante**. El programa contará con el valor predefinido de la gravedad y recibirá los valores de la velocidad, tiempo, coeficiente de arrastre y número de iteraciones, además el rango a y b donde será evaluada la función y con ellos calculará y mostrará las raíces (aproximación a la masa del Paracaidista) y sus posibles errores porcentuales.

FASE 2: ANÁLISIS

Sección de Elementos de Interés

"Técnica de selección o subrayar los sustantivos (elementos de interés)"

Se propone desarrollar un programa que aproxime el valor de la masa de un paracaidista que cae, aplicando el **Método de la Secante**. El programa contará con el valor predefinido de la gravedad y recibirá los valores de la velocidad, tiempo, coeficiente de arrastre y número de iteraciones, además el rango a y b donde será evaluada la función y con ellos calculará y mostrará las raíces (aproximación a la masa del Paracaidista) y sus posibles errores porcentuales.

Lista de Elementos de Interés

- programa
- masa
- Paracaidista
- Método de la Secante
- Valor
- gravedad
- valores
- velocidad
- tiempo
- coeficiente de arrastre
- rango a
- rango b
- raíz
- número de iteraciones

Identificación de los Objetos de Estudio

Método de la Secante Gravedad Velocidad Tiempo

Coeficiente de Arrastre rango a rango b Función Iteraciones

Raíz error relativo porcentual masa_aproximada

Identificación de Composición del Objeto de Estudio

Elementos Significativos

Método_Secante

- Iteración
- Raíces (x_0 y x_1)
- Función_Evaludad(fx0 y fx1)
- Error_R_P

Función_Paracaedista

- Gravedad
- Masa_aproximada (rango a y b)
- Velocidad
- Coeficiente_arrastre
- Tiempo

Determinación de Métodos para la Obtención de Resultados

-Se requiere métodos para colocar valores en los atributos

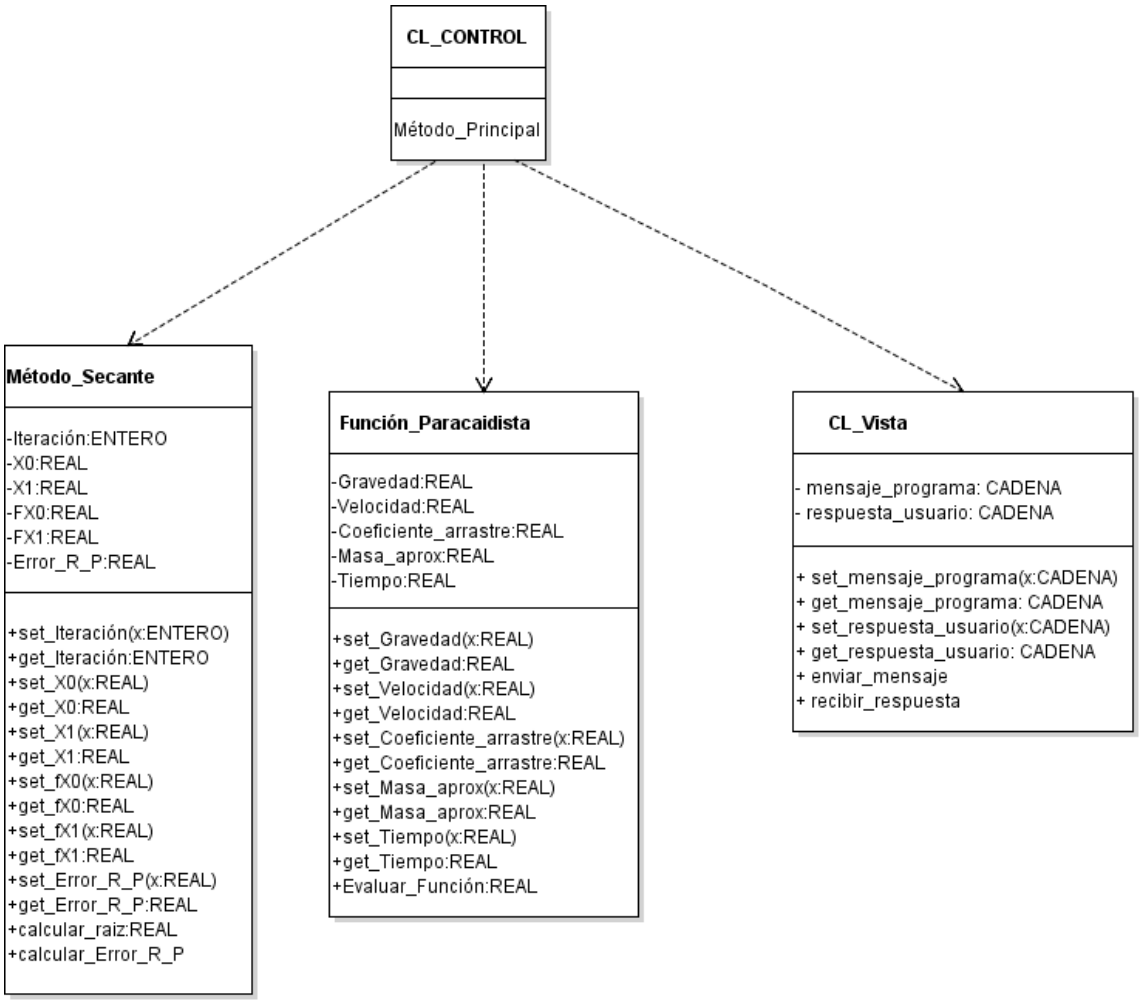
-Se requiere métodos para obtener valores de los atributos

-Se requiere un método en la clase Función_Paracaedista para evaluar la función dada.

-Se requiere un método para calcular las raíces de las aproximaciones de la masa en la clase Método_Secante

-Se requiere un método para calcular el error relativo en la clase Método_Secante

DIAGRAMA DE CLASES



Pseudocódigo

```
CLASE  Método_Secante
INICIO
SECCIÓN DE ATRIBUTOS
 Iteracion:ENTERO,PRIVADO
 X0:REAL,PRIVADO
 X1:REAL,PRIVADO
FX0:REAL,PRIVADO
FX1:REAL:PRIVADO
Error_R_P:REAL,PRIVADO

SECCIÓN DE MÉTODOS
 MÉTODO PUBLICO Método_Secante(ite: ENTERO,x0,x1,fx0,fx1:REAL)
 INICIO
  Iteración← ite
  X0←x0
  X1←x1
  FX0←fx0
  FX1←fx1
 FIN MÉTODO Método_Secante

 MÉTODO PUBLICO set_Iteración(x: ENTERO)
 INICIO
  Iteración← x
 FIN MÉTODO set_Iteración

 MÉTODO PUBLICO get_Iteración: ENTERO
 INICIOI
  REGRESAR Iteración
 FIN MÉTODO get_Iteración

 MÉTODO PUBLICO set_X0(x: REAL)
 INICIO
  X0← x
 FIN MÉTODO set_X0

 MÉTODO PUBLICO get_X0: REAL
 INICIO
  REGRESAR X0
 FIN MÉTODO get_X0

 MÉTODO PUBLICO set_X1(x: REAL)
 INICIO
  X1← x
 FIN MÉTODO set_X1

 MÉTODO PUBLICO get_X1: REAL
 INICIO
  REGRESAR X1
 FIN MÉTODO get_X1

 MÉTODO PUBLICO set_fX0(x: REAL)
 INICIO
  fX0← x
 FIN MÉTODO set_fX0

 MÉTODO PUBLICO get_fX0: REAL
 INICIO
  REGRESAR fX0
 FIN MÉTODO get_fX0

 MÉTODO PUBLICO set_fX1(x: REAL)
 INICIO
  fX1← x
 FIN MÉTODO set_fX1
```

MÉTODO PUBLICO get_fX1: REAL
INICIO
REGRESAR fX1
FIN MÉTODO get_fX1

MÉTODO PUBLICO set_Error_R_P(x: REAL)
INICIO
Error_R_P← x
FIN MÉTODO set_Error_R_P

MÉTODO PUBLICO get_Error_R_P: REAL
INICIO
REGRESAR Error_R_P
FIN MÉTODO get_Error_R_P

MÉTODO calcular_raiz:REAL
Raiz_calculada:REAL
INICIO
Raiz_calculada$\leftarrow x1-(\frac{fx1(x0-x1)}{fx0-fx1})$
Regresar Raiz_calculada
FIN MÉTOTO calcular_raiz

MÉTODO calcular_Error_R_P:
INICIO
Error_R_P$\leftarrow absoluto(\left(\frac{(x1-x0)}{x1}\right)*100\%)$
FIN MÉTOTO calcular_Error_R_P

FIN CLASE Método_Secante

CLASE Función_Paracaedista
INICIO
SECCIÓN DE ATRIBUTOS
Gravedad:REAL,PRIVADO
Velocidad:REAL,PRIVADO
Coeficiente_arrastre:REAL,PRIVADO
Masa_aprox:REAL,PRIVADO
Tiempo:REAL,PRIVADO

SECCIÓN DE MÉTODOS
MÉTODO PUBLICO Función_Paracaedista(g,v,ca,masa: REAL)
INICIO
Gravedad ← g
Velocidad ← v
Coeficiente_arrastre ← ca
Masa_aprox ← masa
FIN MÉTODO Función_Paracaedista

MÉTODO PUBLICO set_Gravedad(x: REAL)
INICIO
Gravedad ← x
FIN MÉTODO set_Gravedad

MÉTODO PUBLICO get_Gravedad: REAL
INICIO
REGRESAR Gravedad
FIN MÉTODO get_Gravedad

MÉTODO PUBLICO set_Velocidad(x: REAL)
INICIO
Velocidad ← x
FIN MÉTODO set_Velocidad

MÉTODO PUBLICO get_Velocidad: REAL
INICIO
REGRESAR Velocidad
FIN MÉTODO get_Velocidad

MÉTODO PUBLICO set_Coeficiente_arrastre(x: REAL)
INICIO
Coeficiente_arrastre ← x
FIN MÉTODO set_Coeficiente_arrastre

MÉTODO PUBLICO get_Coeficiente_arrastre: REAL
INICIO
REGRESAR Coeficiente_arrastre
FIN MÉTODO get_Coeficiente_arrastre

MÉTODO PUBLICO set_Masa_aprox(x: REAL)
INICIO
Masa_aprox ← x
FIN MÉTODO set_Masa_aprox

MÉTODO PUBLICO get_Masa_aprox: REAL
INICIO
REGRESAR Masa_aprox
FIN MÉTODO get_Masa_aprox

MÉTODO PUBLICO set_Tiempo(x: REAL)
INICIO
Tiempo ← x
FIN MÉTODO set_Tiempo

MÉTODO PUBLICO get_Tiempo: REAL
INICIO
REGRESAR Tiempo
FIN MÉTODO get_Tiempo

MÉTODO Evaluar_Función:REAL
Función_Evaluada:REAL
INICIO
Función_Evaluada $\leftarrow\left(\frac{Gravedad*(Masa\_aprox)}{Coeficiente\_arrastre}*\left(1-e^{-\left(\frac{coeficiente\_arrastre}{Masa\_aprox}\right)*tiempo}\right)\right)-$ velocidad
FIN MÉTODO Función_Evaluada

FIN CLASE Función_Paracaidista

CLASE CL_Vista
DESCRIPCIÓN: Clase que interactúa con el Usuario
INICIO
SECCIÓN DE ATRIBUTOS
 mensaje_programa:CADENA,PRIVADO
 respuesta_usuario:CADENA,PRIVADO

SECCIÓN DE MÉTODOS

MÉTODO PÚBLICO set_mensaje_programa(x:CADENA)
 INICIO
  mensaje_programa←x
FIN MÉTODO set_mensaje_programa

MÉTODO PÚBLICO get_mensaje_programa:CADENA
 INICIO
  REGRESAR mensaje_programa
FIN MÉTODO get_mensaje_programa

MÉTODO PÚBLICO set_respuesta_usuario(x:CADENA)
 INICIO
  respuesta_usuario←x
FIN MÉTODO set_respuesta_usuario

MÉTODO PÚBLICO get_respuesta_usuario:CADENA
 INICIO
  REGRESAR respuesta_usuario
FIN MÉTODO get_respuesta_usuario

MÉTODO PÚBLICO enviar_mensaje
INICIO
 Escribe mensaje_programa
FIN MÉTODO enviar_mensaje

MÉTODO PÚBLICO recibir_respuesta
INICIO
 LEE respuesta_usuario
FIN MÉTODO recibir_respuesta

FIN CLASE CL_Vista

```
CLASE CL_Control
INICIO
  SECCIÓN DE VARIABLES
    mensaje1: CADENA
    mensaje2: CADENA
    mensaje3: CADENA
    mensaje4: CADENA
    mensaje5: CADENA
    mensaje6: CADENA
    mensaje7: CADENA
    mensaje8: CADENA
    mensaje9: CADENA
    respuesta:CADENA
    v:REAL
    t:REAL
    c_a:REAL
    a:REAL
    b:REAL
    g=9.81:REAL
    ite:ENTERO
    x0:REAL
    x1:REAL
    fx0:REAL
    fx1:REAL
    Iteraciones:ARREGLO DE ENTEROS
    Raices:ARREGLO DE REALES
    Funciones:ARREGLO DE REALES
    Errores:ARREGLO DE REALES
  SECCIÓN DE METODOS
  MÉTODO PRINCIPAL
  INICIO
    mensaje1← “Método de la Secante”
    mensaje2←”Aproximar la Masa de un Paracaidista ”
    mensaje3←”Ingrese la Velocidad:  ”
    mensaje4←”Ingrese Tiempo: ”
    mensaje5←”Ingrese el Coeficiente de arrastre:  ”
    mensaje6←”Ingrese el Rango a:  ”
    mensaje7←”Ingrese el Rango b: ”
    mensaje8←”Ingrese el Número de Iteraciones:  ”
    mensaje9←”Tabla de las Iteraciones: ”

    objeto_mivista: OBJETO DE CL_Vista
    objeto_mivista.set_mensaje_programa(mensaje1)
    objeto_mivista.enviar_mensaje
    objeto_mivista.set_mensaje_programa(mensaje2)
    objeto_mivista.enviar_mensaje
    objeto_mivista.set_mensaje_programa(mensaje3)
    objeto_mivista.enviar_mensaje
    objeto_mivista.recibir_respuesta
    respuesta←objeto_mivista.get_respuesta_usuario
    v←CONVERTIRA_REAL(respuesta)
```

```
objeto_mivista.set_mensaje_programa(mensaje4)
objeto_mivista.enviar_mensaje
objeto_mivista.recibir_respuesta
respuesta←objeto_mivista.get_respuesta_usuario
t←CONVERTIRA_REAL(respuesta)
objeto_mivista.set_mensaje_programa(mensaje5)
objeto_mivista.enviar_mensaje
objeto_mivista.recibir_respuesta
respuesta←objeto_mivista.get_respuesta_usuario
c_a←CONVERTIRA_REAL(respuesta)
objeto_mivista.set_mensaje_programa(mensaje6)
objeto_mivista.enviar_mensaje
objeto_mivista.recibir_respuesta
respuesta←objeto_mivista.get_respuesta_usuario
a←CONVERTIRA_REAL(respuesta)
objeto_mivista.set_mensaje_programa(mensaje7)
objeto_mivista.enviar_mensaje
objeto_mivista.recibir_respuesta
respuesta←objeto_mivista.get_respuesta_usuario
b←CONVERTIRA_REAL(respuesta)
objeto_mivista.set_mensaje_programa(mensaje8)
objeto_mivista.enviar_mensaje
objeto_mivista.recibir_respuesta
respuesta←objeto_mivista.get_respuesta_usuario
ite←CONVERTIRA_ENTERO(respuesta)

Iteraciones[ite] ARREGLO DE ENTEROS;
Raices[ite] ARREGLO DE REALES;
Funciones[ite] ARREGLO DE REALES;
Errores[ite] ARREGLO DE REALES;

Para i=0 hacer hasta (ite)
        Iteraciones[i]=0
        Raices[i]=0
        Funciones[i]=0
        Errores[i]=0
FIN PARA

PARA i=0 hacer hasta (ite+1)
        SI (i < 2) Entonces
                Si (i==0) Entonces
                        objeto_miFunción: OBJETO DE Función_Paracaedista(g,v,c_a,a,t)
                        Funciones[i]←objeto_miFunción.Evaluar_Función
                        Raices[i]←a
                        Iteraciones[i]←i-1
                        Errores[i]←9999
                Sino
                        objeto_miFunción: OBJETO DE Función_Paracaedista(g,v,c_a,b,t)
                        Funciones[i]←objeto_miFunción.Evaluar_Función
                        Raices[i]←b
```

```
                    Iteraciones[i]←i-1
                    x0=Raices[i-1]
                    x1=Raices[i]
                    fx0=Funciones[i-1]
                    fx1=Funciones[i]
                    Objeto_miSecante:OBJETO DE Método_Sacante(ite,x0,x1,fx0,fx1)
                    objeto_miSecante.calcular_Error_R_P
                    Errores[i]←objeto_miSecante.get_Error_R_P

        SI NO
                Objeto_miSecante:OBJETO DE Método_Sacante(ite,x0,x1,fx0,fx1)
                Raices[i]←Objeto_miSecante.calcular_raiz
                x0←Raices[i-1]
                x1←Raices[i]
                objeto_miFunción: OBJETO DE Función_Paracaedista(g,v,c_a,x1,t)
                Funciones[i]←objeto_miFunción.Evaluar_Función
                fx0←Funciones[i-1]
                fx1←Funciones[i]
                objeto_miSecante.set_X0(x0)
                objeto_miSecante.set_X1(x1)
                objeto_miSecante.calcular_Error_R_P
                Errores[i]←objeto_miSecante.get_Error_R_P
                Iteraciones[i]←i-1
FIN PARA

PARA i=0 hacer hasta (ite-1)
        Escribe ('Iteraciones: ',  Interaciones[i]
                + ' Raices: ', Raices[i]
                + 'fxi: ', Funciones[i]
                + 'Error Relativo Porcentual: ' Errores[i])
FIN PARA


  FIN DE MÉTODO PRINCIPAL
FIN CLASE CL_Control
```

IMPLEMENTACIÓN (Usando el Lenguaje de Programación C++)

Tipo de paradigma: Paradigma de Programación Orientado a Objetos

Codificación del Pseudocódigo

- C++

```
#include <iostream>
#include<stdlib.h>
#include <string>
#include<math.h>
#include <cmath>
#include <iomanip>

using namespace std;
class Funcion_Paracaedista{
 private:
    float Gravedad;
    float Velocidad;
    float Coeficiente_arrastre;
    float Masa_aprox;
    float Tiempo;
 public:
    Funcion_Paracaedista(float,float,float,float,float);
    void set_Gravedad(float x);
    float get_Gravedad();
    void set_Velocidad(float x);
    float get_Velocidad();
    void set_Coeficiente_arrastre(float x);
    float get_Coeficiente_arrastre();
    void set_Masa_aprox(float x);
    float get_Masa_aprox();
    void set_Tiempo(float x);
    float get_Tiempo();
    float Evaluar_Funcion();
};

Funcion_Paracaedista::Funcion_Paracaedista(float g,float v,float ca,float masa,float t){
 Gravedad=g;
 Velocidad=v;
 Coeficiente_arrastre=ca;
 Masa_aprox=masa;
 Tiempo=t;
}
void Funcion_Paracaedista::set_Gravedad(float x){
 Gravedad=x;
}
float Funcion_Paracaedista::get_Gravedad(){
 return Gravedad;
}
void Funcion_Paracaedista::set_Velocidad(float x){
 Velocidad=x;
}
float Funcion_Paracaedista::get_Velocidad(){
 return Velocidad;
}
void Funcion_Paracaedista::set_Coeficiente_arrastre(float x){
 Coeficiente_arrastre=x;
}
float Funcion_Paracaedista::get_Coeficiente_arrastre(){
 return Coeficiente_arrastre;
}
void Funcion_Paracaedista::set_Masa_aprox(float x){
 Masa_aprox=x;
}
float Funcion_Paracaedista::get_Masa_aprox(){
 return Masa_aprox;
}
void Funcion_Paracaedista::set_Tiempo(float x){
 Tiempo=x;
}
```

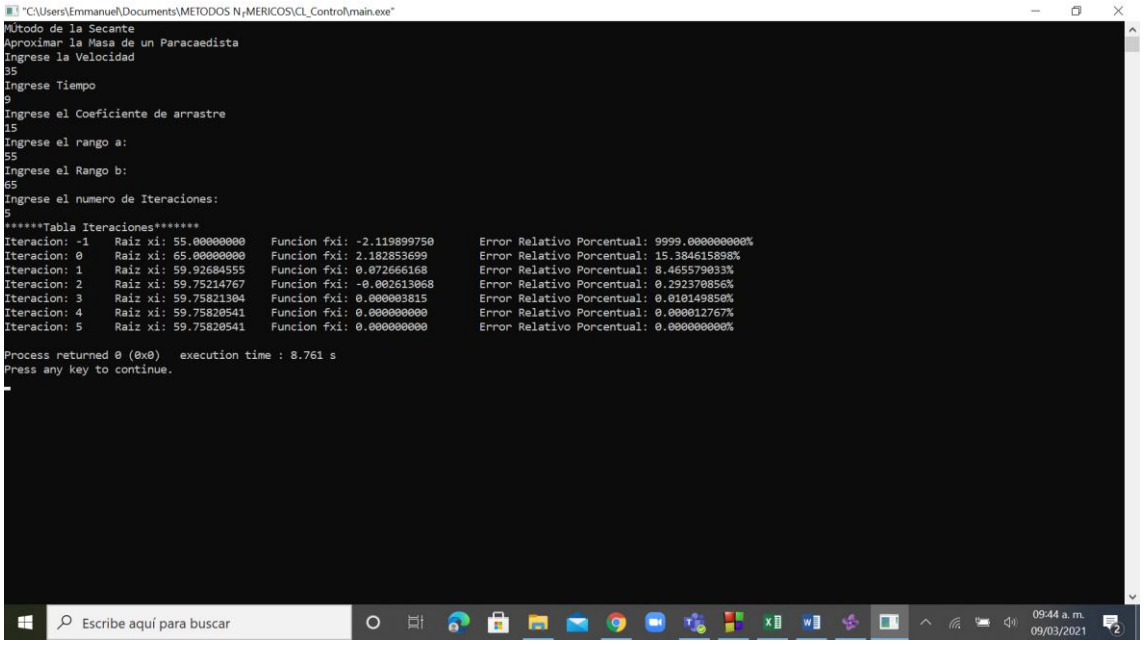
```
float Funcion_Paracaedista::get_Tiempo(){
 return Tiempo;
}
float Funcion_Paracaedista::Evaluar_Funcion(){
 float Funcion_Evaluada;
 Funcion_Evaluada=(((Gravedad*Masa_aprox)/(Coeficiente_arrastre))*(1-exp(-
(Coeficiente_arrastre/Masa_aprox)*Tiempo))-Velocidad);
 return Funcion_Evaluada;
}

class CL_Vista{
 private:
    string mensaje_programa;
    string respuesta_usuario;
 public:
    void set_mensaje_programa(string x);
    string get_mensaje_programa();
    void set_respuesta_usuario(string x);
    string get_respuesta_usuario();
    void enviar_mensaje();
    void recibir_respuesta();
};

void CL_Vista::set_mensaje_programa(string x){
 mensaje_programa=x;
}
string CL_Vista::get_mensaje_programa(){
 return mensaje_programa;
}
void CL_Vista::set_respuesta_usuario(string x){
 respuesta_usuario=x;
}
string CL_Vista::get_respuesta_usuario(){
 return respuesta_usuario;
}
void CL_Vista::enviar_mensaje(){
 cout<<mensaje_programa<<endl;
}
void CL_Vista::recibir_respuesta(){
 cin>>respuesta_usuario;
}

class Metodo_Secante{
 private:
    int Iteracion;
    float X0;
    float X1;
    float FX0;
    float FX1;
    float Error_R_P;
 public:
    Metodo_Secante(int,float,float,float,float);
    void set_Iteracion(int x);
    int get_Iteracion();
    void set_X0(float x);
    float get_X0();
    void set_X1(float x);
    float get_X1();
    void set_fX0(float x);
    float get_fX0();
    void set_fX1(float x);
    float get_fX1();
    void set_Error_R_P(float x);
    float get_Error_R_P();
    float calcular_raiz();
    void calcular_Error_R_P();
};
Metodo_Secante::Metodo_Secante(int ite,float x0,float x1,float fx0,float fx1){
 Iteracion=ite;
 X0=x0;
 X1=x1;
 FX0=fx0;
 FX1=fx1;
}
```

```
void Metodo_Secante::set_Iteracion(int x){
 Iteracion=x;
}
int Metodo_Secante::get_Iteracion(){
 return Iteracion;
}
void Metodo_Secante::set_X0(float x){
 X0=x;
}
float Metodo_Secante::get_X0(){
 return X0;
}
void Metodo_Secante::set_X1(float x){
 X1=x;
}
float Metodo_Secante::get_X1(){
 return X1;
}
void Metodo_Secante::set_fX0(float x){
 FX0=x;
}
float Metodo_Secante::get_fX0(){
 return FX0;
}
void Metodo_Secante::set_fX1(float x){
 FX1=x;
}
float Metodo_Secante::get_fX1(){
 return FX1;
}
void Metodo_Secante::set_Error_R_P(float x){
 Error_R_P=x;
}
float Metodo_Secante::get_Error_R_P(){
 return Error_R_P;
}
float Metodo_Secante::calcular_raiz(){
 float Raiz_calculada;
 Raiz_calculada=(X1-(FX1*(X0-X1))/(FX0-FX1));
 return Raiz_calculada;
}
void Metodo_Secante::calcular_Error_R_P(){
 Error_R_P=abs(((X1-X0)/(X1))*100);
}

int main()
{
   string mensaje1;
   string mensaje2;
   string mensaje3;
   string mensaje4;
   string mensaje5;
   string mensaje6;
   string mensaje7;
   string mensaje8;
   string mensaje9;
   string respuesta;
   float v;
   float t;
   float c_a;
   float a;
   float b;
   float g=9.81;
   int ite;
   float x0;
   float x1;
   float fx0;
   float fx1;


   mensaje1="Método de la Secante";
   mensaje2="Aproximar la Masa de un Paracaedista";
   mensaje3="Ingrese la Velocidad";
   mensaje4="Ingrese Tiempo";
```

```
mensaje5="Ingrese el Coeficiente de arrastre";
mensaje6="Ingrese el rango a: ";
mensaje7="Ingrese el Rango b: ";
mensaje8="Ingrese el numero de Iteraciones: ";
mensaje9="******Tabla Iteraciones*******";

CL_Vista objeto_miVista=CL_Vista();
objeto_miVista.set_mensaje_programa(mensaje1);
objeto_miVista.enviar_mensaje();
objeto_miVista.set_mensaje_programa(mensaje2);
objeto_miVista.enviar_mensaje();
objeto_miVista.set_mensaje_programa(mensaje3);
objeto_miVista.enviar_mensaje();
objeto_miVista.recibir_respuesta();
respuesta=objeto_miVista.get_respuesta_usuario();
v=std::stof(respuesta);
objeto_miVista.set_mensaje_programa(mensaje4);
objeto_miVista.enviar_mensaje();
objeto_miVista.recibir_respuesta();
respuesta=objeto_miVista.get_respuesta_usuario();
t=std::stof(respuesta);
objeto_miVista.set_mensaje_programa(mensaje5);
objeto_miVista.enviar_mensaje();
objeto_miVista.recibir_respuesta();
respuesta=objeto_miVista.get_respuesta_usuario();
c_a=std::stof(respuesta);
objeto_miVista.set_mensaje_programa(mensaje6);
objeto_miVista.enviar_mensaje();
objeto_miVista.recibir_respuesta();
respuesta=objeto_miVista.get_respuesta_usuario();
a=std::stof(respuesta);
objeto_miVista.set_mensaje_programa(mensaje7);
objeto_miVista.enviar_mensaje();
objeto_miVista.recibir_respuesta();
respuesta=objeto_miVista.get_respuesta_usuario();
b=std::stof(respuesta);
objeto_miVista.set_mensaje_programa(mensaje8);
objeto_miVista.enviar_mensaje();
objeto_miVista.recibir_respuesta();
respuesta=objeto_miVista.get_respuesta_usuario();
ite=std::stoi(respuesta);

int Iteraciones[ite];
float Raices[ite];
float Funciones[ite];
float Errores[ite];

for(int i=0;i<ite;i++){
   Iteraciones[i]=0;
   Raices[i]=0;
   Funciones[i]=0;
   Errores[i]=0;
}
for(int i=0;i<=ite+1;i++)
{
   if(i<2){
      if(i==0){
         Funcion_Paracaedista objeto_miFuncion=Funcion_Paracaedista(g,v,c_a,a,t);
         Funciones[i]=objeto_miFuncion.Evaluar_Funcion();
         Raices[i]=a;
         Iteraciones[i]=i-1;
         Errores[i]=0;
      }else{
         Funcion_Paracaedista objeto_miFuncion=Funcion_Paracaedista(g,v,c_a,b,t);
         Funciones[i]=objeto_miFuncion.Evaluar_Funcion();
         Raices[i]=b;
         Iteraciones[i]=i-1;
         x0=Raices[i-1];
         x1=Raices[i];
         fx0=Funciones[i-1];
         fx1=Funciones[i];
         Metodo_Secante objeto_miSecante=Metodo_Secante(ite,x0,x1,fx0,fx1);
         objeto_miSecante.calcular_Error_R_P();
         Errores[i]=objeto_miSecante.get_Error_R_P();
      }
```

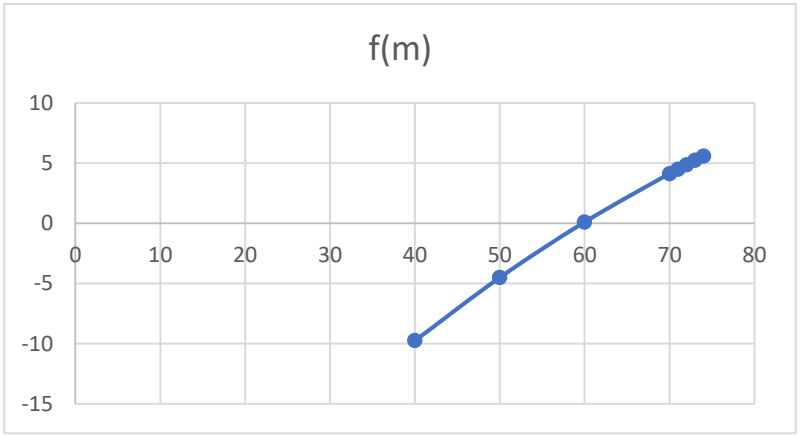
```
        }else{
          Metodo_Secante objeto_miSecante=Metodo_Secante(ite,x0,x1,fx0,fx1);
          Raices[i]=objeto_miSecante.calcular_raiz();
          x0=Raices[i-1];
          x1=Raices[i];
          Funcion_Paracaedista objeto_miFuncion=Funcion_Paracaedista(g,v,c_a,x1,t);
          Funciones[i]=objeto_miFuncion.Evaluar_Funcion();
          fx0=Funciones[i-1];
          fx1=Funciones[i];
          objeto_miSecante.set_X0(x0);
          objeto_miSecante.set_X1(x1);
          objeto_miSecante.calcular_Error_R_P();
          Errores[i]=objeto_miSecante.get_Error_R_P();
          Iteraciones[i]=i-1;
        }
    }
    objeto_miVista.set_mensaje_programa(mensaje9);
    objeto_miVista.enviar_mensaje();

    for(int i=0; i<ite; i++)
    {
      cout<<"Iteracion: "<<Iteraciones[i]<<"\t Raiz xi: "<<fixed<<setprecision(4)<<Raices[i]<<"\t Funcion fxi:
"<<fixed<<setprecision(9)<<Funciones[i]
      <<"\t Error Relativo Porcentual: "<<Errores[i]<<"%" <<endl;
    }
    return 0;
}
```

PROGRAMA CORRIENDO EN CONSOLA



Comprobación

m	f(m)
40	-9.735146
50	-4.4976203
60	0.10413443
70	4.125616
71	4.49870564
72	4.86682132
73	5.23003979
74	5.5884373
75	5.94208953
76	6.29107155
77	6.63545772
78	6.97532164



Método de la Secante			
Iteraciones	xi	fxi	Error_R_P
-1	55	-2.1199033	
0	65	2.18285064	15.38461538
1	59.9268523	0.07266725	8.46556680
2	59.752151	-0.002613	0.29237653
3	59.7582149	2.9807E-06	0.01014744
4	59.758208	1.2206E-10	0.00001156
5	59.758208	0	0.00000000