UNIVERSIDAD NACIONAL
DE ASUNCIÓN

Facultad de
Ingeniería

COMPUTACIÓN

Estructuras en C++

Aux. Marcos Antonio Benítez Ocampos

Resolución de ejercicios

Resolución de ejercicios

En este material se propondrá la resolución de un ejercicio relacionado a la unidad de estructuras en lenguaje de programación C++, así pues, la idea consistirá en que dado el enunciado del tema mencionado anteriormente, proponer entonces en consecuencia una resolución para el mismo.

Es pertinente mencionar que la resolución propuesta no es única, pudiendo los estudiantes encontrar diversas soluciones para el mismo problema y adecuados a su estilo; la resolución propuesta es a criterio del auxiliar de enseñanza una que podría facilitar la comprensión de la utilización de estructuras así como también otros comandos del lenguaje de programación C++.

El ejercicio será detallado lo más posible en este material.

Ejercicio

Utilizando estructuras, elaborar un programa que lea las coordenadas de los puntos p1 y p2 (distintos, no hace falta validar), luego determine:

a) **la pendiente de la recta determinada por p1 y p2;**
b) **la ordenada de un punto cuya abscisa es leída como dato y se encuentra sobre la recta determinada por p1 y p2.**

Como todo programa en lenguaje C++ lo primero es la cabecera, donde incluimos las librerías a utilizar y donde es buena costumbre declarar ya las funciones a utilizar en el programa. Entonces las librerías a utilizar serán:

1.1) Iostream, para el flujo de entrada y salida de datos;
1.2) Cmath, librería utilizada para utilizar funciones matemáticas.

Observaciones:

- Pueden utilizar la librería cmath en vez de math.h, ambas librerías cumplen básicamente con lo mismo, math.h es una librería del lenguaje C propiamente y cmath es de C++.
- Por la versión del compilador que utilizaré para la compilación necesito agregar la librería cstdlib para poder utilizar el comando system("PAUSE"), en otras versiones del compilador esto puede no ser necesario.
- Las librerías cstdlib y stdlib.h se diferencian básicamente en lo mismo que las librerías cmath y math.h.

1.3) Declaramos también la estructura que utilizaremos para poder guardar las coordenadas de los puntos p1 y p2, la estructura tiene el nombre "punto" y tiene dos propiedades "x", "y", que serán utilizadas respectivamente para guardar la abscisa y ordenada de las estructuras de tipo "punto".

1.4) Como tenemos que determinar la pendiente de la recta que pasa por los puntos p1 y p2, entonces crearemos una función de nombre "pendiente". La función entonces recibe como parámetros de entrada los puntos p1 y p2 y retorna un valor de tipo **float**.

1.5) Como tenemos que determinar la ordenada de un punto de la recta que pasa por p1 y p2 y cuya abscisa es conocida, entonces crearemos una función de nombre "ordenada". La función entonces recibe como parámetros de entrada los puntos p1 y p2 y la abscisa conocida, y retorna un valor de tipo **float**.

1.6) También creamos una función que se encargará de leer las coordenadas de los puntos que ingresemos, esta función no tiene parámetros de entrada, pero devuelve valores de tipo "punto". La función se llama "lect".

Entonces, nuestras primeras líneas de programación son las siguientes:

```
#include<iostream>
using namespace std;
#include<cstdlib>
#include<cmath>
struct punto
{
        float x;
        float y;
};
punto lect();
float pendiente( punto p1, punto p2 );
float ordenada( punto p1, punto p2, float x );
```

1.7) Seguidamente pasamos a realizar la función **main**; declaramos la variable **m** de tipo **float** (utilizada para la almacenar el valor de la pendiente) y también las variables p1, p2 y p3 de tipo **punto.** Luego

procedemos a la lectura de las variables de tipo **punto.** Accediendo a cada una de sus propiedades “x”, “y” utilizando el operador “.” Imprimimos finalmente cuales son los puntos ingresados; y calculamos la pendiente con la función “pendiente”.

```
int main()
{
   float m;
   punto p1, p2, p3;
   p1 = lect();
   cout << "\n\np1(" << p1.x << ", " << p1.y << ")\n";
   p2 = lect();
   cout << "\n\np2(" << p2.x << ", " << p2.y << ")\n";
   m = pendiente( p1, p2 );
   cout << "\n\tPendiente de la recta p1, p2 = " << m;
```

1.8) A continuación hay que ingresar el valor de la abscisa del punto p3 para así calcular posteriormente su ordenada correspondiente utilizando la función “ordenada”. Y luego imprimimos finalmente el punto p3.

```
   cout << "\n\n\tDigite la abscisa del punto p ";   cin >> p3.x;
   p3.y = ordenada( p1, p2, p3.x );
   cout << "\n\n\tPunto sobre la recta p3(" << p3.x << ", " << p3.y << ")";
   cout << "\n";
   system ( "pause" );
   return 0;
}
```

Observaciones:

- Notar que para acceder a las propiedades de una estructura se utiliza el operador “.”, en nuestros ejemplos, al utilizar las estructuras p1, p2 y p2, al ser todas estructuras “punto”, todas tienen dos propiedades “x”, “y”.

1.9) Finalmente nos resta escribir el cuerpo de la funciones declaradas. La primera función “lect” es la que se encarga de hacer la lectura de las variables de tipo “punto”, entonces declaramos una variable de tipo “punto” en la función y solicitamos al usuario que introduzca las coordenadas de dicha variable, almacenamos dichos valores en las propiedades de “p” y finalmente retornamos “p”.

```
punto lect()
{
        punto p;
        cout << "\n\tDigite la abscisa del punto:  ";     cin >> p.x;
        cout << "\tDigite la ordenada del punto: ";      cin >> p.y;
        return p;
}
```

1.10) La segunda función “pendiente” es la que se encarga de determinar la pendiente de la recta que pasa por p1 y p2. Entonces declaramos una variable de tipo float en la función que contendrá el valor de la pendiente y calculamos el valor de dicha pendiente con las coordenadas de los puntos que recibe. Hay que tener en cuenta que si la recta es vertical, entonces el valor de la pendiente tiende a infinito por lo que de darse este caso asignamos a la variable “m” un valor tan grande como sea posible.

```
float pendiente( punto p1, punto p2 )
{
        float m;
```

```
        if ( p2.x == p1.x )
            m = 3.4e+38;
        else
            m = (p2.y - p1.y)/(p2.x - p1.x);
        return m;
}
```

1.11) La tercera y última función “ordenada” es la que se encarga de determinar la ordenada de un punto de la recta que pasa por p1 y p2 dada su abscisa. Entonces declaramos una variable de tipo float en la función que contendrá el valor de la ordenada mencionada y calculamos el valor de dicha variable con las coordenadas de los puntos que recibe la función. Hay que tener en cuenta que si la recta es vertical, entonces el valor de la pendiente tiende a infinito por lo que de darse este caso asignamos a la variable “y” un valor tan grande como sea posible.

```
float ordenada( punto p1, punto p2, float x )
{
        float y;
            if ( p2.x == p1.x )
                y = 3.4e+38;
            else
                y = p1.y + (x - p1.x)/(p2.x - p1.x)*(p2.y - p1.y);
            return y;
}
```

Observación:

- Para el prototipado de una función en C++ se tienen dos opciones, declarar la función en la cabecera y luego escribir el cuerpo de la función después de la función **main** (que es lo que nosotros implementamos) o bien en la cabecera declarar la función y escribir su cuerpo también. Generalmente los programadores tienden a programar con el estilo elegido en nuestro código fuente, pues al revisar un código fuente se espera revisar primeramente la función principal y luego las funciones implementadas. Se aclara que ambos estilos son correctos.

Finalmente nuestro código fuente completo es el siguiente:

```
#include<iostream>
using namespace std;
#include<cstdlib>
#include<cmath>
struct punto
{
        float x;
        float y;
};
punto lect();
float pendiente( punto p1, punto p2 );
float ordenada( punto p1, punto p2, float x );
int main()
{
   float m;
   punto p1, p2, p3;
   p1 = lect();
   cout << "\n\np1(" << p1.x << ", " << p1.y << ")\n";
   p2 = lect();
   cout << "\n\np2(" << p2.x << ", " << p2.y << ")\n";
   m = pendiente( p1, p2 );
   cout << "\n\tPendiente de la recta p1, p2 = " << m;
   cout << "\n\n\tDigite la abscisa del punto p ";   cin >> p3.x;
   p3.y = ordenada( p1, p2, p3.x );
   cout << "\n\n\tPunto sobre la recta p3(" << p3.x << ", " << p3.y << ")";
   cout << "\n";
   system ( "pause" );
   return 0;
}
punto lect()
{
        punto p;
        cout << "\n\tDigite la abscisa del punto:  ";     cin >> p.x;
        cout << "\tDigite la ordenada del punto: ";     cin >> p.y;
        return p;
}
float pendiente( punto p1, punto p2 )
{
        float m;
        if ( p2.x == p1.x )
             m = 3.4e+38;
        else
             m = (p2.y - p1.y)/(p2.x - p1.x);
        return m;
}
float ordenada( punto p1, punto p2, float x )
{
        float y;
            if ( p2.x == p1.x )
                 y = 3.4e+38;
            else
                 y = p1.y + (x - p1.x)/(p2.x - p1.x)*(p2.y - p1.y);
            return y;
}
```