SOLUCIONES ACTIVIDAD 9NP

1) Leer dos puntos P1 y P2 representados como registros, calcular la longitud del segmento que los une y la pendiente de la recta que pasa por dichos puntos.

$$longitud = \sqrt{(x1 - x2)^2 + (y1 - y2)^2}$$

$$pendiente = \frac{y2 - y1}{x2 - x1}$$

Definimos un tipo registro con dos campos que representen las coordenadas: x, y respectivamente de un punto en el plano.

La entrada se realiza en un procedimiento de lectura que se invoca cada vez que leemos un punto. Tanto la longitud como la pendiente se determinan en funciones que tienen como entrada los dos puntos y devuelven la longitud y la pendiente respectivamente. La ejecución del programa termina cuando los dos puntos son el mismo.

```
#include <iostream>
#include <math.h>
using namespace std;

typedef struct{
  int x;
  int y;
}TPunto;

TPunto lee();
float longitud (TPunto p1, TPunto p2);
float pendiente (TPunto p1, TPunto p2);
bool distinto(TPunto p1, TPunto p2);

int main(){
  TPunto p1, p2;
  float pend;

  do{
    cout << "LECTURA PRIMER PUNTO" << endl;
    p1=lee();
```

```
    cout << "LECTURA SEGUNDO PUNTO" << endl;
    p2=lee();
    if (distinto(p1,p2)){
      cout << "longitud segmento " << longitud (p1,p2) <<
endl;
      pend=pendiente (p1,p2);
      if (pend!=100)
   cout << "pendiente segmento " << pend << endl;
      else
   cout << "no se puede calcular la pendiente de un segmento
vertical" << endl;
    }
  }while (distinto(p1,p2));
  return 0;
}

TPunto lee(){
  TPunto punto;

  cout << "Introduce la componente x: ";
  cin >> punto.x;
  cout << "Introduce la componente y: ";
  cin >> punto.y;
  return (punto);
}

float longitud (TPunto p1, TPunto p2){
  return (sqrt(pow(p1.x-p2.x,2)+pow(p1.y-p2.y,2)));
}

//si x2==x1 no se puede calcular la pendiente
float pendiente (TPunto p1, TPunto p2){
  float res;

  if (p1.x!=p2.x)
    res = (p2.y-p1.y)/(p2.x-p1.x);
```

```
  else
    res= 100; //valor de error;
  return (res);
}

bool distinto(TPunto p1, TPunto p2){
  bool res;

  if (p1.x==p2.x && p1.y==p2.y)
    res=false;
  else
    res=true;
  return (res);
}
```

2) ¿Qué estructura de datos emplearías para almacenar los siguientes datos de 90 personas? Defínela.
   - Nombre
   - Apellido1
   - Apellido2
   - Número de DNI
   - Fecha de expedición del DNI
   - Fecha de validez del DNI
   - Nombre del padre
   - Apellido 1 del padre
   - Apellido 2 del padre
   - Nombre de la madre
   - Apellido 1 de la madre
   - Apellido 2 de la madre
   - Ciudad de nacimiento
   - Provincia de nacimiento
   - Ciudad donde reside
   - Provincia donde reside
   - Hasta un total de 5 hijos:
     - Nombre de cada hijo
     - Fecha de nacimiento de cada hijo
     - Ciudad de nacimiento de cada hijo
     - Provincia de nacimiento de cada hijo

Fecha de nacimiento

```
typedef char TCadena[15];

typedef struct{
  int dia;
  int mes;
  int anyo;
}TFecha;

typedef struct{
  TCadena nombre;
  TCadena ape1lido1;
  TCadena apellido2;
}TIdentidad;

typedef struct{
  int num[8];
  char letra;
}TDNI;

typedef struct{
  TCadena nombre;
  TFecha nac;
  TCadena ciuNac;
  TCadena provNac;
}THijo;

typedef struct{
  TIdentidad ident;
  TDNI dni;
  TFecha expDNI;
  TFecha valDNI;
  TIdentidad padre;
  TIdentidad madre;
  TCadena ciuNac;
  TCadena provNac;
  TCadena ciuRes;
  TCadena provRes;
```

```
  THijo hijos[5];
  int numHijos;
  TFecha nac;
}TPersona;

typedef TPersona TDatos[90];
```