PARCIAL – ALSE – PUNTO 4 - DESCRIPCIÓN:

NOMBRES: Andrés Rivera Y Santiago Quintero

FECHA: 18/09

Introducción:

El documento describe el funcionamiento del programa solicitado en el punto 4, cuya función es la de calcular la mayor magnitud entre un conjunto de puntos dados.

El usuario tiene la opción de ingresar las coordenadas de los puntos manualmente o de usar un conjunto predeterminado, con dichos puntos calcula la magnitud de cada punto usando la fórmula magnitud ($\sqrt{(x^2 + y^2)}$) , identifica el punto con la mayor magnitud y muestra los resultados.

Funciones y estructuras:

Estructura Point

La estructura Point define un punto en un plano 2D con dos coordenadas:

- double x: Coordenada en el eje x.
- double y: Coordenada en el eje y.
- La estructura se utiliza para almacenar y manipular las coordenadas de los puntos a lo largo del programa.

```
struct Point { // Define la estructura de un Point
    double x, y;
};
```

Función leerpPuntos:

Esta función permite ingresar las coordenadas de los puntos o leer las coordenadas predeterminadas por el programa, se compone de la siguiente manera:

1) Pregunta al usuario si desea ingresar los puntos manualmente (s/n) y lee la respuesta.
    a. Si la respuesta es inválida, entra en un bucle while que solicita una entrada válida (s, S, n, o N).

    b. Si la respuesta es s o S, lee n puntos en formato (x, y) usando std::getline y stringstream, validando el formato con paréntesis y coma. Si el formato es incorrecto, repite la iteración.

c. Si la respuesta es n o N, asigna puntos predeterminados ((0, 0), (3, 4), (6, 8), (9, 12)) hasta n o 4.

Como se observa se hace use de una variable char llamada respuesta como un input entregado por el usuario, para hacer el análisis si lo que entrega es correcto para el proceso del programa, para esto se hace uso ciclos for y while, condicionales if y operaciones booleanas como || (OR) o && (AND).

```
void leerPuntos(Point puntos[], int n) {  // Función para leer las
coordenadas de varios puntos
    char respuesta;
    std::cout << "\n ¿Desea ingresar los puntos manualmente? (s/n): ";
    std::cin >> respuesta;
    std::cin.ignore(); // Ignorar el salto de línea


    if (respuesta != 's' && respuesta != 'S' && respuesta != 'n' &&
respuesta != 'N'){ // Caso en el que el comando no sea sí o no
      while (respuesta != 's' && respuesta != 'S' && respuesta != 'n' &&
respuesta != 'N') {
         std::cout << "\n Respuesta inválida, ingrese s para sí o n para no:
";
         std::cin >> respuesta;
         std::cin.ignore(std::numeric_limits<std::streamsize>::max(), '\n');
// Limpiar el buffer
      }


    }


    if (respuesta == 's' || respuesta == 'S') { // Caso en el que el comando
sea sí
        for (int i = 0; i < n; i++) {
            std::string input;
            std::cout << "\n Ingrese las coordenadas del punto " << i + 1 <<
" (x, y): ";
            std::getline(std::cin, input);


            std::stringstream ss(input);
            char paren1, comma, paren2;
            ss >> paren1 >> puntos[i].x >> comma >> puntos[i].y >> paren2;


            if (paren1 != '(' || comma != ',' || paren2 != ')') {
```

```
                std::cout << "\n Error: El formato debe ser (x, y). Intente
de nuevo.\n";
                i--; // Repetir la iteración
                continue;
            }
        }
    } if (respuesta == 'n' || respuesta == 'N') { // Caso en el que el
comando sea no
        std::cout << "\n Usando puntos predeterminados...\n";
        Point predeterminados[] = {{0, 0}, {3, 4}, {6, 8}, {9, 12}};
        for (int i = 0; i < n && i < 4; i++) {
            puntos[i] = predeterminados[i]; //Asignando los valores de
predeterminado en puntos
        }
    }
}
```

Función calcularMagnitud

La función calcula la magnitud del vector asociado a un punto dado utilizando la fórmula de magnitud $\sqrt{(x^2 + y^2)}$, para esto hace los siguientes pasos en solo una línea de código

1) Recibe un parámetro p1 de tipo const Point& para evitar copias innecesarias y garantizar que no se modifique el punto original.

2) Utiliza pow(p1.x, 2) y pow(p1.y, 2) para elevar al cuadrado las coordenadas, suma estos valores y aplica sqrt para obtener la magnitud.

3) Devuelve el valor de la magnitud como un double.

```
double calcularMagnitud(const Point& p1) { // Función para calcular la
magnitud asociada al punto i
    return sqrt(pow(p1.x, 2) + pow(p1.y, 2));
}
```

Función calcularMayorMagnitud

Determina la mayor magnitud y su índice, siguiendo los siguientes pasos:

1) Inicializa m_my con la magnitud del primer punto y indiceMayorMagnitud en 0.

2) Itera sobre los puntos restantes, calcula la magnitud de cada uno y actualiza m_my y indiceMayorMagnitud si encuentra una magnitud mayor.

3) Devuelve la mayor magnitud encontrada.

Esta función hace uso del ciclo for para iterar sobre cada punto ingresado por el usuario, de esta forma llamando a la función “Calcular Magnitud”, tomar dicho valor y compararlo con la mayor magnitud registrada.

```
double calcularMayorMagnitud(Point puntos[], int n, int& indiceMayorMagnitud)
{  // Función para calcular la mayor magnitud y su índice
    double m_my = calcularMagnitud(puntos[0]);
    indiceMayorMagnitud = 0;


    for (int i = 1; i < n; i++) {
        double m = calcularMagnitud(puntos[i]); // Calcula la magnitud del i
- elemento de puntos
        if (m > m_my) {
            m_my = m; // Compara la magnitud con la mayor magnitud
registrada
            indiceMayorMagnitud = i; // Asigna el valor i a la posición del
punto con mayor magnitud registrada
        }
    }
    return m_my;
}
```

Función mostrarResultado:

Imprime el número del punto (índice + 1), sus coordenadas y la magnitud asociada, usando el índice proporcionado.

```
void mostrarResultado(Point puntos[], int indiceMayorMagnitud, double m_my)
{ // Función para mostrar el punto con mayor magnitud, sus coordenadas y su
magnitud


    std::cout << "\n El punto de mayor magnitud es el " <<
indiceMayorMagnitud + 1 << ", con coordenadas: ("<<
puntos[indiceMayorMagnitud].x << ", "
              << puntos[indiceMayorMagnitud].y << "), el valor de la
magnitud es: " << m_my << std::endl;
}
```

Se hace uso de std::cout simplemente para escribir en la consola, y mostrarle al usuario el resultado bota el programa. En específico, el punto con la mayor magnitud, sus coordenadas y la mayor magnitud.

Función main

Main genera las siguientes acciones durante su ejecucción:

1) Solicita y valida que n sea un entero positivo mayor a 1, manejando entradas inválidas con un mensaje de error. (n siendo la cantidad de puntos en el conjunto evaluado por el programa)
    a. Comprueba primero que sea un número, es decir no un carácter

        i. De no ser un número suelta un mensaje de comando incorrecto limpia la entrada cin y vuelve a solicitarle al usuario de ingresar un número, por medio de asignar n = -1.

    b. Comprueba ahora que el número sea un entero positivo mayor 1
        i. De no ser un entero positivo el programa suelta un mensaje de error y solicita al usuario a ingresar un valor correcto

2) Reserva memoria dinámicamente para n puntos, esto con el fin de almacenar el conjunto de puntos definido anteriormente.

3) Llama a leerPuntos, calcularMayorMagnitud y mostrarResultado en secuencia.
4) Libera la memoria y termina el programa.

Se usa el ciclo do – while para la primera parte del código de main, posterior a eso se asigna o leen los valores a el conjunto “puntos” , se encuentra el máximo valor, se imprimen las respuestas en la consola y se limpia el vector que contiene el conjunto de puntos.

```
int main() {
    int n;


    do {
        std::cout << "Ingrese el número de puntos (mínimo 2): "; // Inicia
el programa
        if (!(std::cin >> n)) { // Caso en el que el comando no sea un
entero positivo
            std::cout << "\n Comando inválido, debe ser un número entero
positivo.\n";
            std::cin.clear();
            std::cin.ignore(std::numeric_limits<std::streamsize>::max(),
'\n');
            n = -1; // Forzar que el bucle continúe
        } else if (n <= 1) {  // Caso en el que el comando no sea mayor a 1
elemento
```

```
            std::cout << "\n Se necesitan al menos 2 puntos para calcular
las magnitudes (recuerde que son enteros positivos).\n";
        }
    } while (n <= 1);


    Point* puntos = new Point[n];


    leerPuntos(puntos, n);


    int indiceMayorMagnitud;
    double m = calcularMayorMagnitud(puntos, n, indiceMayorMagnitud);


    mostrarResultado(puntos, indiceMayorMagnitud, m);


    delete[] puntos;
    return 0;
}
```