LAPORAN PRAKTIKUM

GRAFIKA KOMPUTER

(Dosen : *Rio Priantama S.T., M.T.I.*)

Modul 4

Nama : Muhammad Rizal Nurfirdaus

NIM : 20230810088

Kelas : TINFC-2023-04

TEKNIK INFORMATIKA

FAKULTAS ILMU KOMPUTER

UNIVERSITAS KUNINGAN

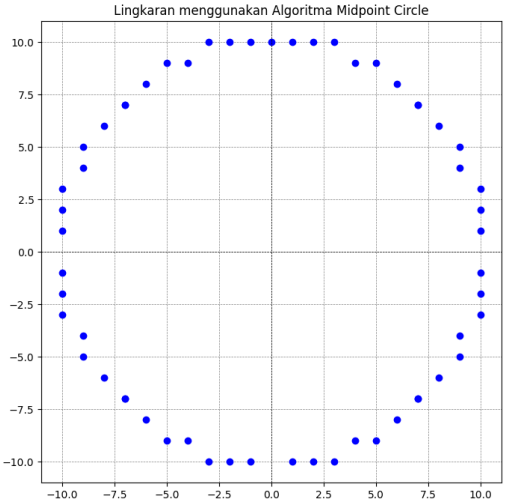
Praktikum

1. Praktikum 1 : Menggambar Lingkaran dengan Algoritma Midpoint
   Source Code:

```
# Praktikum 1
import matplotlib.pyplot as plt

def midpoint_circle(h, k, r):
    x = 0
    y = r
    points = []

    # Parameter keputusan
    p = 1 - r

    # Menggambar titik di delapan kuadran (titik awal)
    points.append((h + x, k + y))

    while x < y:
        x += 1
        if p < 0:
            p = p + 2 * x + 1
        else:
            y -= 1
            p = p + 2 * x - 2 * y + 1

        # Menambahkan titik dari simetri (oktan pertama)
        points.extend([
            (h + x, k + y),
            (h - x, k + y),
            (h + x, k - y),
            (h - x, k - y),
            (h + y, k + x),
            (h - y, k + x),
            (h + y, k - x),
            (h - y, k - x)
        ])

    return points

# Parameter lingkaran
center_x = 0
center_y = 0
radius = 10

# Menggambar lingkaran
circle_points = midpoint_circle(center_x, center_y, radius)

# Menyiapkan plot
plt.figure(figsize=(8, 8))

for point in circle_points:
    # Menggambar titik lingkaran
```

```
    plt.plot(point[0], point[1], 'bo')

plt.title('Lingkaran menggunakan Algoritma Midpoint Circle')
plt.xlim(-radius - 1, radius + 1)
plt.ylim(-radius - 1, radius + 1)
plt.axhline(0, color='black', linewidth=0.5, ls='--')
plt.axvline(0, color='black', linewidth=0.5, ls='--')
plt.grid(color='gray', linestyle='--', linewidth=0.5)
plt.gca().set_aspect('equal', adjustable='box')
plt.show()
```

Hasil Run:



Analisis: Program Kode & Tujuan Kode menggunakan Algoritma Midpoint Circle untuk menghasilkan koordinat titik-titik diskrit yang mendekati lingkaran berpusat di (h,k) dengan jari-jari r. Algoritma menghitung titik pada satu oktaf (octant) menggunakan parameter keputusan p dan merefleksikannya ke tujuh simetri lainnya sehingga satu perhitungan menghasilkan delapan titik pada setiap iterasi.

Kebenaran & logika Pembaruan parameter keputusan (p = p + 2*x + 1 saat p < 0, dan p = p + 2*x - 2*y + 1 setelah y -= 1 saat p >= 0) sesuai dengan bentuk umum algoritma (biasanya ditulis p = p + 2(x - y) + 1 pada cabang negatif). Loop while x < y juga lazim dipakai untuk menghentikan ketika oktaf sudah lengkap; alternatif while x <= y kadang digunakan untuk memastikan titik diagonal (x == y) juga disertakan. Secara keseluruhan perhitungan inti tampak benar dan kompleksitasnya O(r).

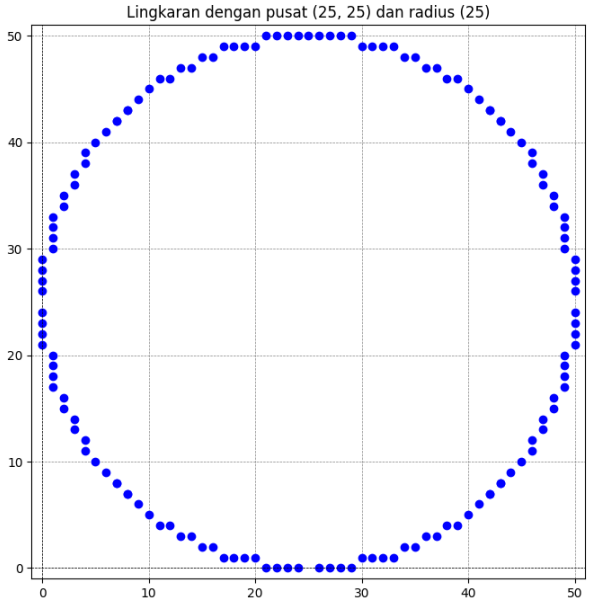
Pengamatan pada Plot menunjukkan lingkaran diskrit berjari 10 dengan simetri yang jelas di ke-8 kuadran; titik-titik membentuk keliling lingkaran tetapi tampak sedikit berkerumun/duplikat pada beberapa posisi (karena setiap iterasi menambah 8 titik tanpa pengecekan duplikat, terutama dekat awal dan saat x ≈ y). Kepadatan titik konsisten per integer step pada x dan y sehingga terlihat “titik-titik” bukan garis kontinu itu memang sifat algoritma raster.

2. Praktikum 2 : Menggambar Lingkaran dengan Algoritma Simetri
   Source Code:

```
# Prakitum 2
import matplotlib.pyplot as plt
```

```python
def midpoint_circle(h, k, r):
    x = 0
    y = r
    points = []

    p = 1 - r

    points.append((h + x, k + y))

    while x < y:
        x += 1
        if p < 0:
            p = p + 2 * x + 1
        else:
            y -= 1
            p = p + 2 * x - 2 * y + 1

        # Menambahkan simetri lingkaran
        points.extend([
            (h + x, k + y),
            (h - x, k + y),
            (h + x, k - y),
            (h - x, k - y),
            (h + y, k + x),
            (h - y, k + x),
            (h + y, k - x),
            (h - y, k - x)
        ])

    return points

# Input dari pengguna
center_x = int(input("Masukkan nilai koordinat x pusat lingkaran: "))
center_y = int(input("Masukkan nilai koordinat y pusat lingkaran: "))
radius = int(input("Masukkan nilai radius lingkaran: "))

# Menghitung titik-titik lingkaran menggunakan Algoritma Midpoint Circle
circle_points = midpoint_circle(center_x, center_y, radius)

# Menyiapkan plot
plt.figure(figsize=(8, 8))

for point in circle_points:
    plt.plot(point[0], point[1], 'bo') # Menggambar titik lingkaran

plt.title(f"Lingkaran dengan pusat ({center_x}, {center_y}) dan radius
({radius})")
plt.xlim(center_x - radius - 1, center_x + radius + 1)
plt.ylim(center_y - radius - 1, center_y + radius + 1)
plt.axhline(0, color='black', linewidth=0.5, ls='--')
plt.axvline(0, color='black', linewidth=0.5, ls='--')
plt.grid(color='gray', linestyle='--', linewidth=0.5)
plt.gca().set_aspect('equal', adjustable='box')
plt.show()
```

Hasil Run:



Analisis: Program Kode pada Praktikum 2 ini merupakan implementasi dari Algoritma Midpoint Circle dengan penambahan fitur input dari pengguna untuk menentukan pusat dan jari-jari lingkaran secara dinamis. Algoritma ini menggunakan pendekatan *incremental decision parameter* untuk menentukan titik-titik koordinat yang membentuk keliling lingkaran berdasarkan simetri delapan bagian (*octant symmetry*), sehingga perhitungan menjadi lebih efisien tanpa perlu menggunakan fungsi trigonometri atau akar kuadrat.

Hasil plot menampilkan lingkaran berpusat di titik (25, 25) dengan jari-jari 25. Titik-titik biru membentuk pola melingkar yang simetris di semua kuadran, menunjukkan bahwa perhitungan algoritma berjalan dengan benar. Garis bantu koordinat dan grid membantu memperjelas posisi relatif lingkaran terhadap sumbu X dan Y. Lingkaran terlihat halus dan seimbang karena setiap iterasi menambahkan delapan titik sekaligus yang merepresentasikan simetri lingkaran.

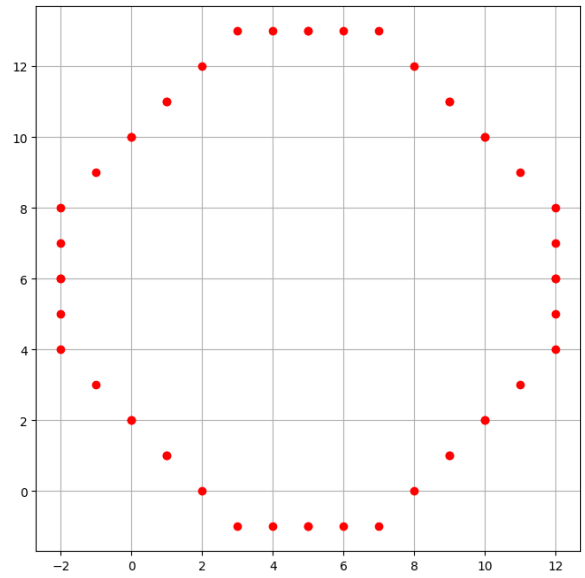
Secara keseluruhan, program ini sudah berjalan baik dan sesuai konsep algoritma Midpoint Circle. Visualisasi yang dihasilkan menunjukkan keberhasilan algoritma dalam menggambar lingkaran dengan ketelitian tinggi. Untuk peningkatan, dapat ditambahkan penghindaran duplikasi titik agar hasil lebih efisien, serta penggunaan plt.scatter() agar tampilan titik lebih konsisten.

3. Praktikum 3 : Menggambar Lingkaran dengan Algoritma Bresenhem
   Source Code:

```
# Praktikum 3
import matplotlib.pyplot as plt

def plot_circle(x_center, y_center, x, y):
    # Menggambar semua delapan simetri lingkaran
    plt.plot(x_center + x, y_center + y, 'ro')
    plt.plot(x_center - x, y_center + y, 'ro')
    plt.plot(x_center + x, y_center - y, 'ro')
```

```
    plt.plot(x_center - x, y_center - y, 'ro')
    plt.plot(x_center + y, y_center + x, 'ro')
    plt.plot(x_center - y, y_center + x, 'ro')
    plt.plot(x_center + y, y_center - x, 'ro')
    plt.plot(x_center - y, y_center - x, 'ro')

def bresenham_circle(x_center, y_center, radius):
    x = 0
    y = radius

    # Initial decision parameter d = 3 - 2 * radius
    d = 3 - 2 * radius

    plot_circle(x_center, y_center, x, y)

    while y >= x:
        x += 1

        if d > 0:
            y -= 1
            d = d + 4 * (x - y) + 10
        else:
            d = d + 4 * x + 6

        plot_circle(x_center, y_center, x, y)

# Input dinamis dari pengguna
x_center = int(input("Masukkan koordinat pusat lingkaran (x_center): "))
y_center = int(input("Masukkan koordinat pusat lingkaran (y_center): "))
radius = int(input("Masukkan jari-jari lingkaran (radius): "))

# Menyiapkan plot
plt.figure(figsize=(8, 8))

# Memanggil fungsi algoritma Bresenham untuk menggambar lingkaran
bresenham_circle(x_center, y_center, radius)

# Menampilkan lingkaran yang digambar
plt.gca().set_aspect('equal', adjustable='box')
plt.grid(True)
plt.show()
```

Hasil Run :



Analisis: Program pada Praktikum 3 ini menggunakan Algoritma Bresenham Circle untuk menggambar lingkaran dengan efisien tanpa operasi pecahan maupun akar kuadrat. Prinsip algoritma ini adalah menghitung titik-titik pada satu oktan (bagian 1/8 dari lingkaran) menggunakan parameter keputusan d = 3 - 2r, lalu memanfaatkan sifat simetri lingkaran untuk menggambar tujuh titik lainnya yang memiliki jarak sama terhadap pusat lingkaran.

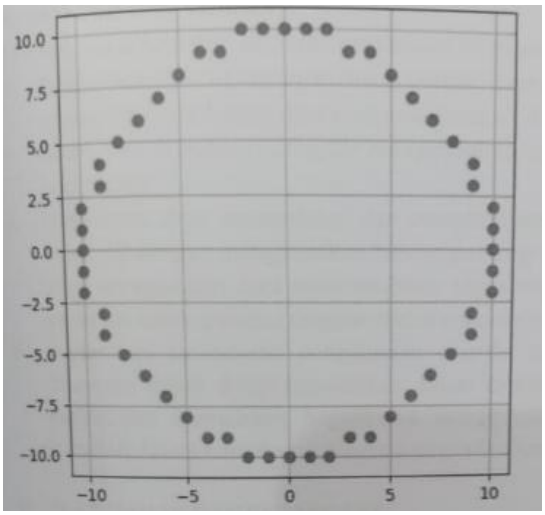
Secara logika, algoritma dimulai dengan titik awal (x=0, y=r). Nilai d menentukan apakah titik berikutnya tetap di baris yang sama atau turun satu piksel (mengurangi nilai y). Jika d > 0, maka y dikurangi satu dan d diperbarui dengan rumus d = d + 4(x - y) + 10. Sebaliknya, jika d <= 0, maka y tetap dan d diperbarui dengan d = d + 4x + 6. Proses ini berulang hingga x melampaui y, menandakan satu oktan sudah selesai.

Hasil plot menunjukkan lingkaran berwarna merah dengan bentuk yang simetris dan pusat sesuai input pengguna. Titik-titik tersusun rapi membentuk keliling lingkaran, menandakan bahwa algoritma bekerja dengan benar dan presisi tinggi. Grid membantu memperjelas posisi setiap titik terhadap koordinat pusat.

Secara keseluruhan, implementasi dan hasil visualisasi sudah benar. Program ini berhasil menggambarkan lingkaran menggunakan algoritma Bresenham secara efisien dan akurat. Untuk peningkatan visual, dapat ditambahkan label judul dan sumbu agar hasil lebih informatif.

LATIHAN/TUGAS

1. Buatlah kode program sederhana untuk menggambar lingkaran seperti gambar berikut:

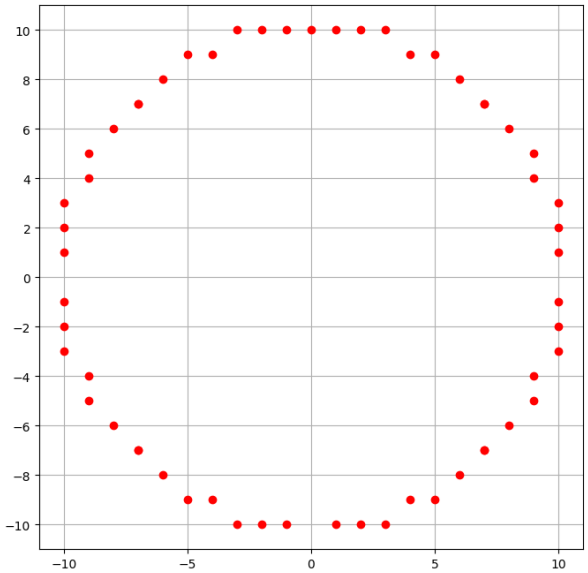


Source Code:

```
# Tugas
import matplotlib.pyplot as plt

def midpoint_circle(h, k, r):
    """
    Mengimplementasikan Algoritma Midpoint Circle untuk menghitung
    titik-titik lingkaran di delapan oktan.
    """
    x = 0
    y = r
    points = []

    # Parameter keputusan awal p = 1 - r
    p = 1 - r

    # Menambahkan titik awal di (h+x, k+y)
    points.append((h + x, k + y))

    while x < y:
        x += 1
        if p < 0:
            p = p + 2 * x + 1
        else:
            y -= 1
            p = p + 2 * x - 2 * y + 1

        # Menambahkan titik dari simetri (8 oktan)
        points.extend([
            (h + x, k + y),
            (h - x, k + y),
            (h + x, k - y),
            (h - x, k - y),
            (h + y, k + x),
```

```
            (h - y, k + x),
            (h + y, k - x),
            (h - y, k - x)
        ])

    return points

# --- Parameter Lingkaran sesuai Gambar ---
center_x = 0
center_y = 0
radius = 10

# Menghitung titik-titik lingkaran
circle_points = midpoint_circle(center_x, center_y, radius)

# --- Menyiapkan Plot ---
plt.figure(figsize=(8, 8))

for point in circle_points:
    # Menggambar titik lingkaran ('bo' = Blue Dot)
    plt.plot(point[0], point[1], 'o', color='red')

# Pengaturan sumbu agar sama dengan gambar
plt.xlim(-radius - 1, radius + 1) # Batas x dari -11 hingga 11
plt.ylim(-radius - 1, radius + 1) # Batas y dari -11 hingga 11
plt.xticks(range(-10, 11, 5)) # Label sumbu x
plt.yticks(range(-10, 11, 2)) # Label sumbu y
plt.grid(True)
plt.gca().set_aspect('equal', adjustable='box')
plt.show()
```

Hasil Run :



Analisis: Program pada Tugas ini menggunakan Algoritma Midpoint Circle untuk menggambar lingkaran dengan pusat di koordinat (0,0) dan jari-jari 10. Algoritma ini bekerja dengan menghitung titik-titik yang membentuk lingkaran menggunakan perhitungan integer, tanpa operasi trigonometri atau akar kuadrat, sehingga efisien dan cepat untuk diaplikasikan pada sistem raster.

Proses dimulai dari titik awal di atas lingkaran (x=0, y=r) dengan parameter keputusan awal p = 1 - r. Setiap iterasi, nilai x bertambah satu, sementara y dapat berkurang tergantung hasil evaluasi nilai p. Jika $p < 0$, titik berikutnya tetap pada baris yang sama; jika $p \geq 0$, maka y dikurangi satu. Hasil tiap iterasi digunakan untuk membentuk delapan titik simetri lingkaran (8 oktan) agar bentuknya tetap bulat sempurna di semua kuadran.

Hasil visualisasi menunjukkan lingkaran berwarna merah yang simetris dan mulus, terbentuk dari kumpulan titik-titik diskrit yang mengelilingi pusat koordinat. Tampilan plot juga disertai grid dan pengaturan sumbu yang proporsional (aspect ratio = equal), sehingga lingkaran tampak seimbang dan tidak terdistorsi.

Secara keseluruhan, implementasi dan hasil visualisasi sudah sangat baik. Program berhasil menggambarkan lingkaran secara akurat sesuai prinsip Midpoint Circle Algorithm, menunjukkan efisiensi perhitungan dan ketepatan simetri yang tinggi.