

**PRE TEST DAN POST TEST PRAKTIKUM  
GRAFIKA KOMPUTER**  
**(Dosen : Rio Priantama S.T., M.T.I.)**

**Modul 3**



**Nama : Muhammad Rizal Nurfirdaus**

**NIM : 20230810088**

**Kelas : TINFC-2023-04**

**TEKNIK INFORMATIKA  
FAKULTAS ILMU KOMPUTER  
UNIVERSITAS KUNINGAN**

## PRE-TEST

1. Jelaskan perbedaan utama antara algoritma DDA dan Bresenham dalam cara menghitung koordinat piksel pada layar!
    - Metode perhitungan:
      - DDA (Digital Differential Analyzer) menghitung titik piksel dengan menggunakan pembagian dan angka riil (floating-point). Ia menghitung  $X_{inc} = dx/step$  dan  $Y_{inc} = dy/step$  lalu menambahkan inkremen riil ke posisi saat ini setiap langkah, kemudian melakukan pembulatan untuk memilih piksel.
      - Bresenham menggunakan *integer arithmetic* dan parameter keputusan (error term) sehingga tidak perlu pembagian atau bilangan riil. Perubahan posisi dibuat dengan operasi penjumlahan dan perbandingan integer.
    - Akurasi & performa:
      - DDA lebih sederhana konsepnya tapi lebih mahal secara komputasi karena operasi floating point dan pembulatan tiap langkah.
      - Bresenham lebih cepat dan efisien (khususnya di hardware lama) karena hanya operasi integer — biasanya hasilnya juga visualnya lebih “rapat” tanpa akumulasi kesalahan floating.
    - Penggunaan:
      - DDA mudah dipahami/diimplementasikan (baik untuk pembelajaran).
      - Bresenham umum dipakai pada rasterisasi garis di grafik karena efisiensi dan presisi integer-nya.
    - Perilaku pada kemiringan besar/kecil:
      - Keduanya menangani semua kuadran dan kemiringan; Bresenham mengatur langkah berdasarkan error untuk memilih sumbu yang diinkrement, sedangkan DDA selalu menambah  $X_{inc}$  dan  $Y_{inc}$  (proporsional).
2. Jika titik awal ( $x_0, y_0$ ) adalah (2, 3) dan titik akhir ( $x_1, y_1$ ) adalah (10, 8), hitung nilai  $dx$ ,  $dy$ , dan  $step$  yang digunakan dalam algoritma DDA!  
Diberi:  $x_0 = 2$ ,  $y_0 = 3$ ,  $x_1 = 10$ ,  $y_1 = 8$ .
    - $dx = x_1 - x_0 = 10 - 2 = 8$ .
    - $dy = y_1 - y_0 = 8 - 3 = 5$ .
    - $step = \max(|dx|, |dy|) = \max(8, 5) = 8$ .Maka inkremen per langkah:
    - $X_{inc} = dx/step = 8/8 = 1.0$ .
    - $Y_{inc} = dy/step = 5/8 = 0.625$ .Jika mulai dari (2,3) dan setiap langkah menambahkan (1.0, 0.625) lalu membulatkan untuk menentukan piksel, deret koordinat (dibulatkan ke bilangan bulat terdekat) adalah:
    - 1) i=0: (2.000, 3.000) → (2, 3)
    - 2) i=1: (3.000, 3.625) → (3, 4)
    - 3) i=2: (4.000, 4.250) → (4, 4)
    - 4) i=3: (5.000, 4.875) → (5, 5)
    - 5) i=4: (6.000, 5.500) → (6, 6)

- 6) i=5: (7.000, 6.125) → (7, 6)
- 7) i=6: (8.000, 6.750) → (8, 7)
- 8) i=7: (9.000, 7.375) → (9, 7)
- 9) i=8: (10.000, 8.000) → (10, 8)

(Perhatikan: biasanya loop dijalankan step kali dan tiap iterasi menggambar titik hasil pembulatan; daftar di atas termasuk titik awal dan titik akhir.)

### POST-TEST

1. Berikan contoh penerapan Algoritma Bresenham untuk menggambar garis dari titik (3, 2) ke titik (10, 5). Tuliskan hasil koordinat setiap pixel yang diperoleh dari proses algoritma tersebut.

```
# Post Test 1

import matplotlib.pyplot as plt

# --- Algoritma Bresenham ---

# Titik awal dan akhir
x0, y0 = 3, 2
x1, y1 = 10, 5

# Hitung perbedaan
dx = x1 - x0
dy = y1 - y0

# Inisialisasi parameter keputusan
p = 2 * dy - dx
x, y = x0, y0

# Simpan titik pertama
points = [(x, y)]

# Looping algoritma Bresenham
for i in range(dx):
    if p < 0:
        x += 1
        p += 2 * dy
    else:
        x += 1
        y += 1
        p += 2 * (dy - dx)
    points.append((x, y))

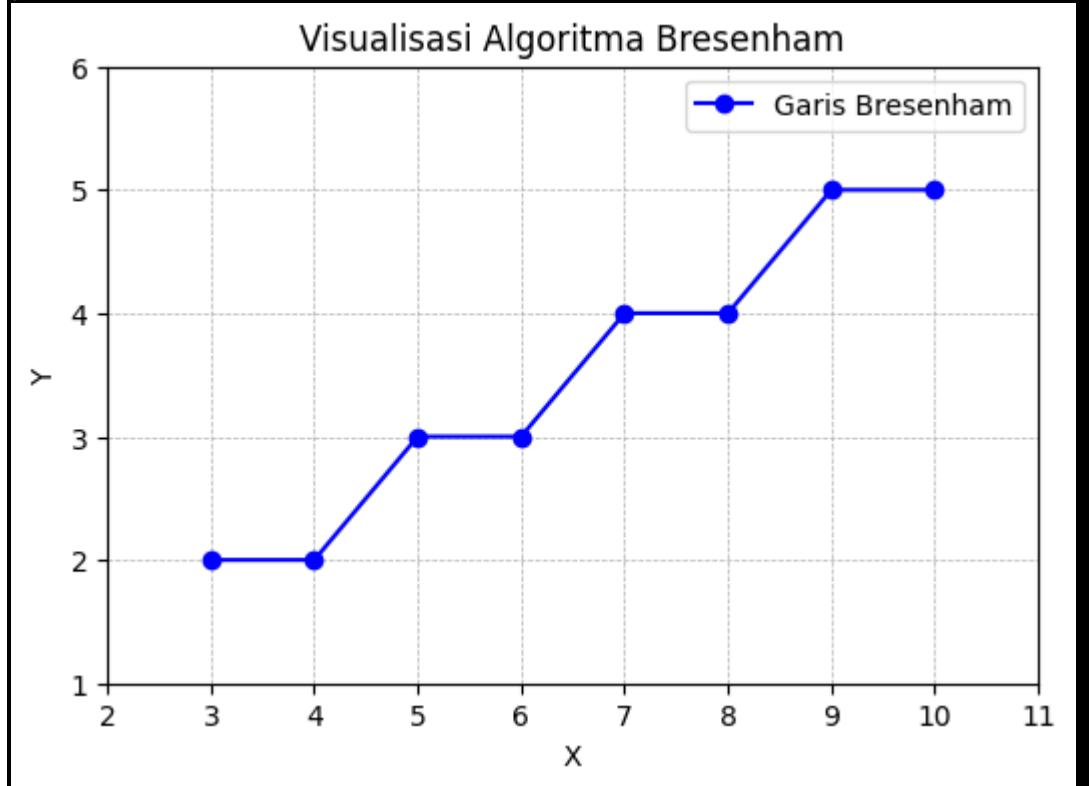
# Cetak hasil koordinat pixel
print("Koordinat setiap pixel hasil Algoritma Bresenham:")
for pt in points:
    print(pt)
```

```

# --- Visualisasi hasil ---
# Pisahkan koordinat x dan y
x_coords, y_coords = zip(*points)

# Buat plot titik pixel
plt.figure(figsize=(6, 4))
plt.plot(x_coords, y_coords, 'bo-', label='Garis Bresenham') # 'bo-' = bulatan biru
plt.grid(True, which='both', linestyle='--', linewidth=0.5)
plt.title('Visualisasi Algoritma Bresenham')
plt.xlabel('X')
plt.ylabel('Y')
plt.xticks(range(min(x_coords)-1, max(x_coords)+2))
plt.yticks(range(min(y_coords)-1, max(y_coords)+2))
plt.legend()
plt.show()

```



Koordinat setiap pixel hasil Algoritma Bresenham:

- (3, 2)
- (4, 2)
- (5, 3)
- (6, 3)
- (7, 4)
- (8, 4)
- (9, 5)
- (10, 5)

2. Pada aplikasi pemrograman grafis, kapan lebih tepat menggunakan Algoritma DDA, dan kapan lebih tepat menggunakan Algoritma Bresenham? Berikan contoh kasus penggunaannya masing-masing!  
contoh kasusnya seperti perbandingan Algoritma DDA dan Bresenham menggunakan bahasa Python, lengkap dengan visualisasi grafik.

```
#Post Test 2
import matplotlib.pyplot as plt

# Titik awal dan akhir
x0, y0 = 3, 2
x1, y1 = 10, 5

# =====
# ◇ ALGORITMA DDA
# =====
def dda_line(x0, y0, x1, y1):
    points = []
    dx = x1 - x0
    dy = y1 - y0
    step = max(abs(dx), abs(dy))
    Xinc = dx / step
    Yinc = dy / step

    x, y = x0, y0
    for i in range(int(step) + 1):
        points.append((round(x), round(y)))
        x += Xinc
        y += Yinc
    return points

# =====
# ◇ ALGORITMA BRESENHAM
# =====
def bresenham_line(x0, y0, x1, y1):
    points = []
    dx = abs(x1 - x0)
    dy = abs(y1 - y0)
    p = 2 * dy - dx
    x, y = x0, y0

    # Tentukan arah
    sx = 1 if x1 > x0 else -1
    sy = 1 if y1 > y0 else -1

    for i in range(dx + 1):
        points.append((x, y))
        if p < 0:
            y += sy
            p += 2 * dy
        else:
            x += sx
            p -= 2 * dx
```

```

        if p >= 0:
            y += sy
            p += 2 * (dy - dx)
        else:
            p += 2 * dy
        x += sx
    return points

# =====
# ◇ HASIL KOORDINAT
# =====
dda_points = dda_line(x0, y0, x1, y1)
bres_points = bresenham_line(x0, y0, x1, y1)

print("Koordinat DDA:")
print(dda_points)
print("\nKoordinat Bresenham:")
print(bres_points)

# =====
# ◇ VISUALISASI GRAFIK
# =====
plt.figure(figsize=(7, 4))
# DDA (merah)
x_dda, y_dda = zip(*dda_points)
plt.plot(x_dda, y_dda, 'ro--', label='DDA Line')

# Bresenham (biru)
x_bre, y_bre = zip(*bres_points)
plt.plot(x_bre, y_bre, 'bo-', label='Bresenham Line')

plt.title('Perbandingan Algoritma DDA vs Bresenham')
plt.xlabel('X')
plt.ylabel('Y')
plt.grid(True, linestyle='--', linewidth=0.5)
plt.legend()
plt.xticks(range(min(x0, x1) - 1, max(x0, x1) + 2))
plt.yticks(range(min(y0, y1) - 1, max(y0, y1) + 2))
plt.show()

```

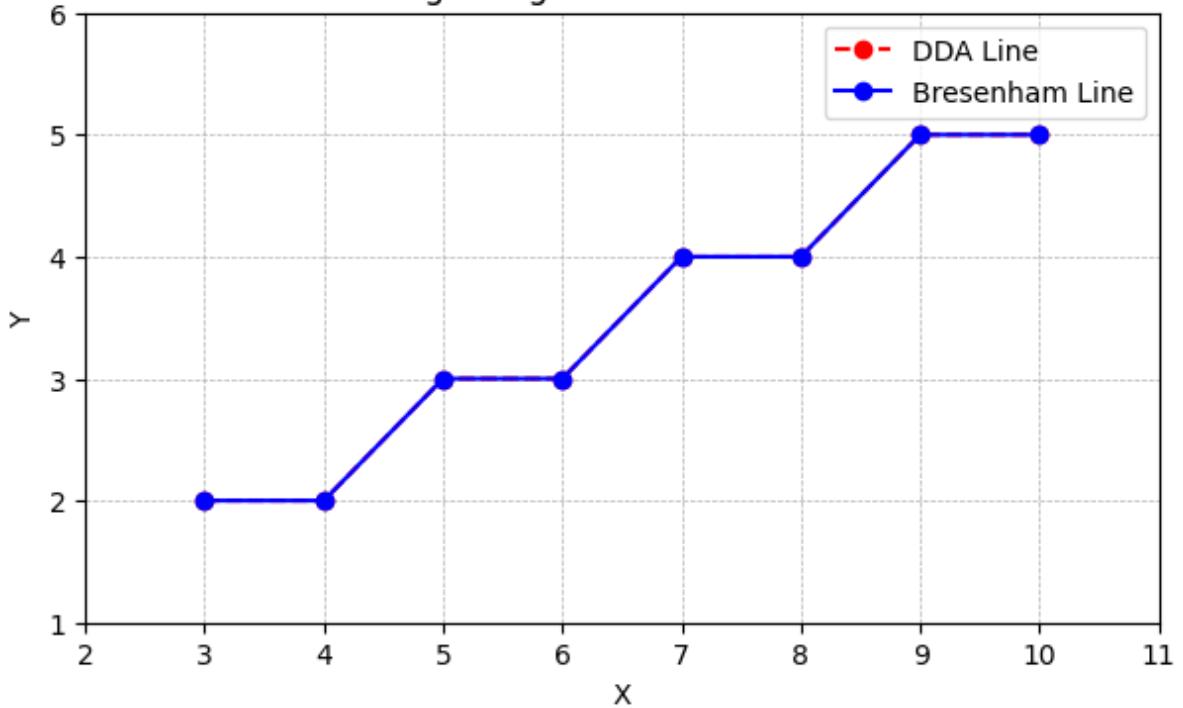
Koordinat DDA:

`[(3, 2), (4, 2), (5, 3), (6, 3), (7, 4), (8, 4), (9, 5), (10, 5)]`

Koordinat Bresenham:

`[(3, 2), (4, 2), (5, 3), (6, 3), (7, 4), (8, 4), (9, 5), (10, 5)]`

### Perbandingan Algoritma DDA vs Bresenham



Pada contoh kasus di atas, algoritma DDA dan Bresenham sama-sama digunakan untuk menggambar garis dari titik (3, 2) ke (10, 5), namun tujuan penggunaannya berbeda. Algoritma DDA lebih cocok dipakai dalam konteks pembelajaran atau aplikasi grafis sederhana karena menggunakan perhitungan bilangan pecahan (float) yang mudah dipahami, sehingga sering dipakai untuk demonstrasi konsep interpolasi titik pada bidang koordinat. Sebaliknya, algoritma Bresenham lebih efisien digunakan pada sistem grafis nyata seperti game 2D, mikrokontroler, atau perangkat keras grafis karena hanya memakai operasi bilangan bulat (integer) yang lebih cepat dan akurat tanpa kesalahan pembulatan. Dalam praktiknya, DDA ideal untuk simulasi dan visualisasi edukatif, sedangkan Bresenham menjadi pilihan utama pada aplikasi yang membutuhkan kecepatan dan presisi tinggi dalam menggambar garis pada layar piksel.