```
// DDA によるディジタル直線の高速描画プログラム dda.c の断片

// 直線を描く関数（低速版，直接解法）
// x1, y1：始点の座標
// x2, y2：終点の座標
// c：文字
void SlowDrawLine(int x1, int y1, int x2, int y2, char c)
{
        double  m;
        int     dx, dy;
        int     x, y;

        dx = x2 - x1;
        dy = y2 - y1;
        m = (double)dy/(double)dx;
        for (x = 0; x <= dx; x++) {
        //      y = (int)round(m*x);    // round() は四捨五入 <math.h>
                y = m*x + 0.5;          // 正の数ならこれでも同じだし速い
                Plot(x1 + x, y1 + y, c);
        }
}


// 直線を描く関数（高速版）
// 実数版DDA（インクリメンタル計算の基本形）
void FastDrawLineDDA1(int x1, int y1, int x2, int y2, char c)
{
        double  m;
        int     dx, dy;
        int     x, y;
        double  e;

        dx = x2 - x1;                   // 線分のx方向の長さ
        dy = y2 - y1;                   // 線分のy方向の長さ
        m = (double)dy/(double)dx;      // 線分の傾き（微分値）
        e = 0.0;                        // y座標の誤差
        y = 0;
        for (x = 0; x <= dx; x++) {
                Plot(x1 + x, y1 + y, c);
                e += m; // 誤差に微分値を加算
                        // !! 実数計算の誤差蓄積により直線が歪む可能性あり
                if (e >= 0.5) {         // 誤差が0.5以上なら...
                        y++;                    // y座標を増やす
                        e -= 1.0;               // 誤差は減らす
                }
        }
}
```

```
// 整数版DDA（誤差蓄積を解消，速度も向上）
//  （実数版の数式を全体的に dx 倍し，すべての計算を整数化）
void FastDrawLineDDA2(int x1, int y1, int x2, int y2, char c)
{
//      double  m;              // 実数は使わない
        int     dx, dy;
        int     x, y;
//      double  e;              // 同上
        int     e;              // 整数化した誤差

        dx = x2 - x1;
        dy = y2 - y1;
        e = 0;
        y = 0;
        for (x = 0; x <= dx; x++) {
                Plot(x1 + x, y1 + y, c);
                e += dy;        // dy = m*dx
                if (2*e >= dx) {
                        y++;
                        e -= dx;
                }
        }
}
```