



세종대학교

Report

윈도우 프로그래밍 4장 연습문제 7, 심화문제 1

제출일 2018.05.21

전공 디지털콘텐츠학과

학번 16013093

이름 박상우

VS 2018 버전을 사용했습니다.

연습문제 7

먼저 좌표계는 GetClientRect()함수로 화면의 가로, 세로 크기를 저장해 아래와 같은 코드로 구현했습니다.

```
// 수직선을 그립니다.
dc.MoveTo(rect.Width() / 2 , 0);
dc.LineTo(rect.Width() / 2, rect.Height());
// 수평선을 그립니다.
dc.MoveTo(0, rect.Height() / 2);
dc.LineTo(rect.Width(), rect.Height() / 2);
```

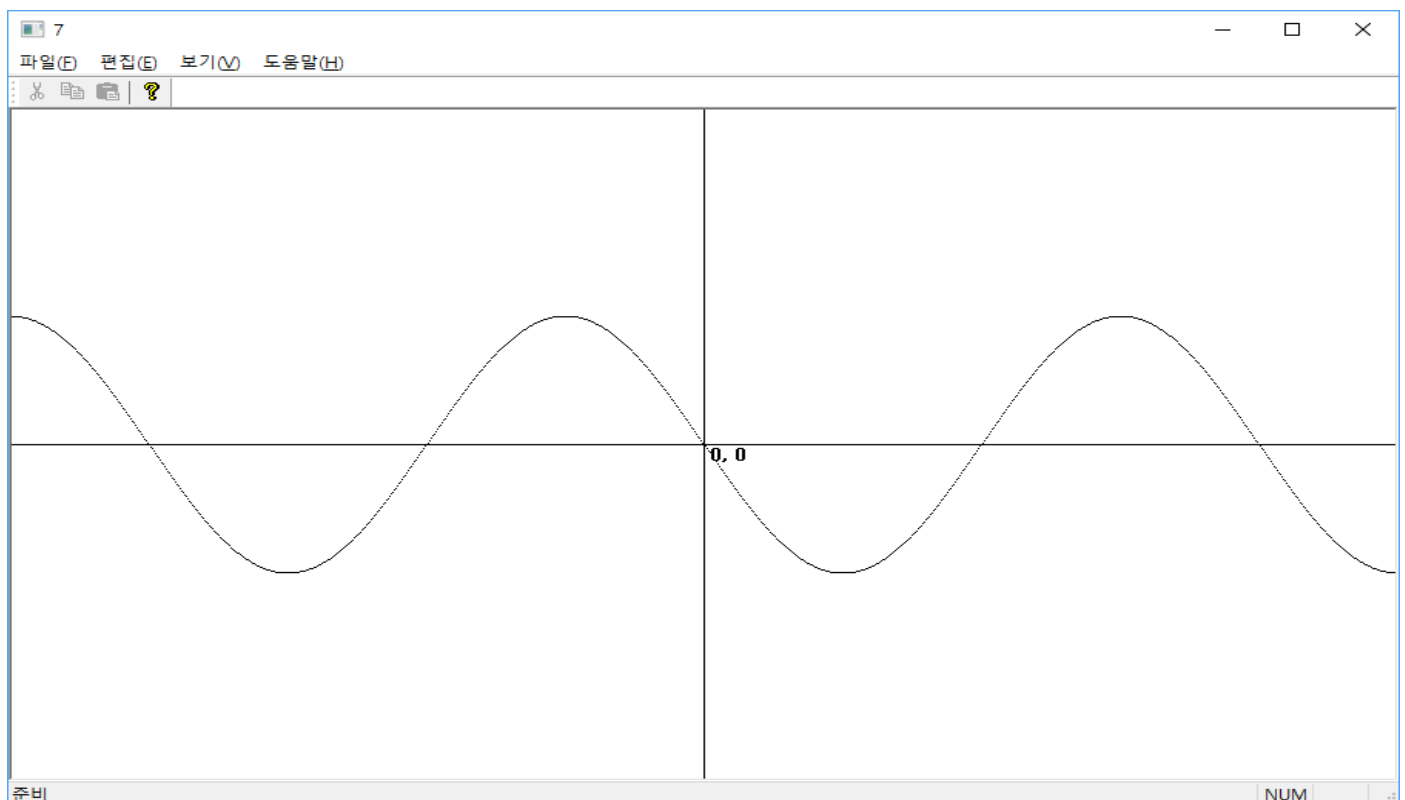
이 후 sin()함수를 이용해 x,y 좌표를 구하고 SetPixelV() 함수를 이용해 점을 찍어가며 출력했습니다.
사인값을 구하는 함수는 double sin(double x)입니다.

x는 라디안 값이고 1rad은 180/3.14 degree입니다. 때문에 $x * 3.14 / 180$ 로 좌표를 변환 하여 사인함수의 파라미터로 사용했습니다.

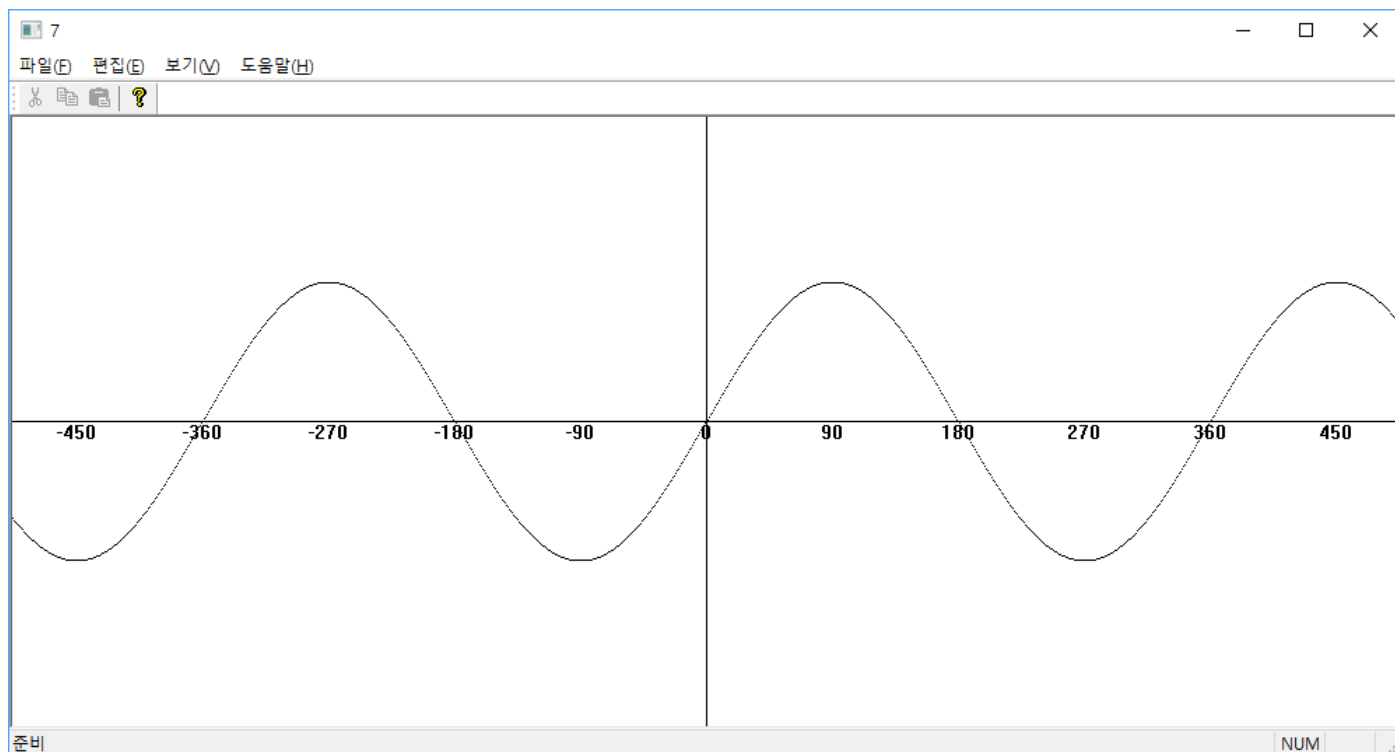
```
for(int x = -rect.Width() / 2; x < rect.Width() / 2; x++)
{
    y = sin(x * 3.14/180) ;
    dc.SetPixelV(x + rect.Width() / 2, (int)(y * 100) + rect.Height() / 2 , RGB(0, 0, 0));
}
```

x의 좌표는 화면 정중앙이 0이 되게 설정했습니다. 때문에 x의 범위는 $-rect.Width() / 2 \sim rect.Width() / 2$ 까지이며 SetPixelV()를 사용 할 때도 화면 높이의 절반 만큼을 더해 실제 클라이언트의 주소로 바꿔 화면 크기를 조절해도 항상 화면 정 중앙에 나타낼 수 있게 했습니다.

또한 항상 sin의 값은 -1~1사이로 이 값을 사용해 점을 찍는다면 int x, int y를 파라미터로 받는 SetPixelV() 함수에서 y값은 항상 0으로만 찍히게 되고 맨 윗줄만 그려지게 됩니다. 때문에 100배를 해 비율을 키운 뒤 화면 높이의 절반만큼 더해 화면 정중앙이 중심이 되게 했습니다.

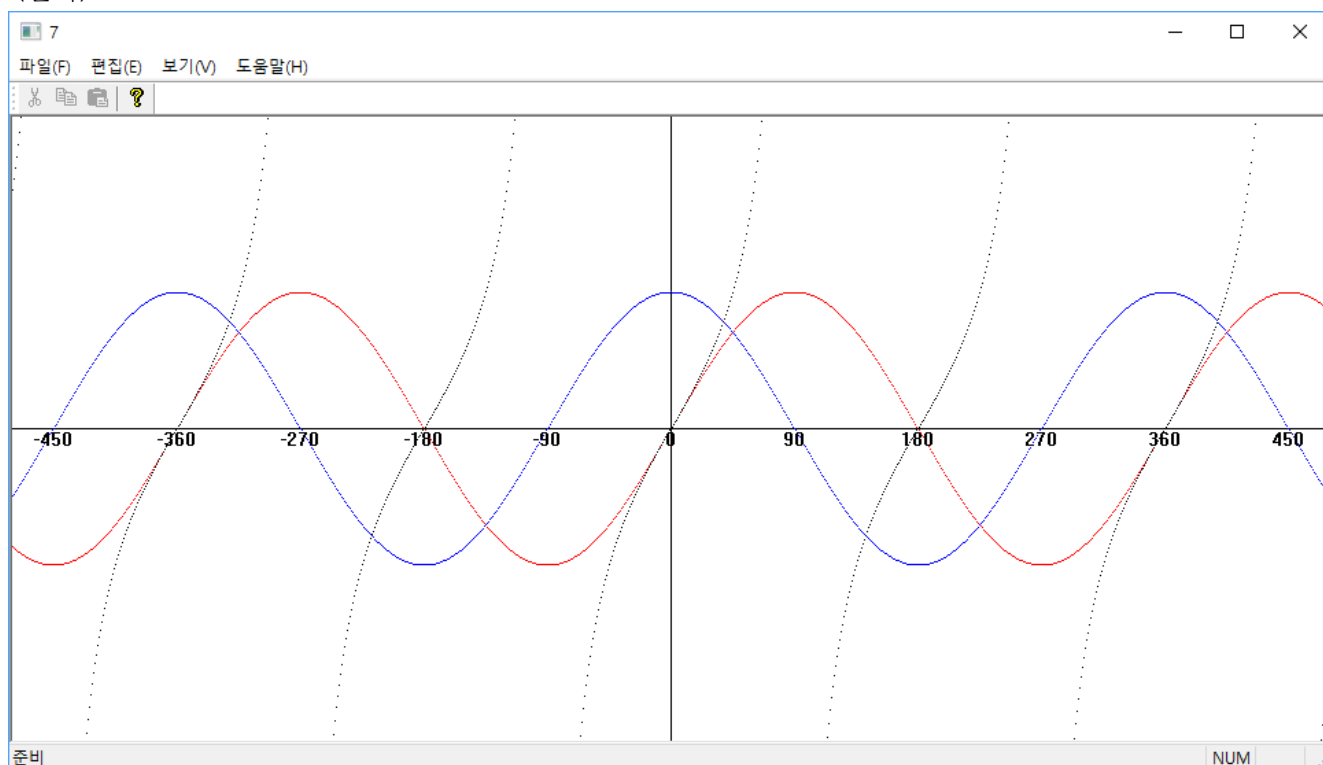


출력을 하니 그래프가 뒤집혀 출력되었고, 클라이언트의 좌표계는 y축의 경우 맨 위가 0부터 시작하기 때문에 sin그래프가 뒤집혀 나온 것이라고 판단해 -100을 곱하는 방법으로 해결했습니다.
 추가로 화면 정중앙을 기준으로 90씩 증감하며 좌표를 찍어 제대로 그려졌는지도 확인했습니다.



같은 방법으로 cos함수와 tan함수도 그려보았습니다. x를 1씩 증가시키는 경우 점이 드문드문 찍히는 부분을 확인했는데, x는 1씩 증가하지만 삼각함수 값을 y축으로 100배 키우면서 빈 공간이 생기는 것으로 생각됩니다. x를 double형으로 바꾸고 0.1씩 증가시켜봤지만 화면 크기 조절 시 너무 느리게 화면이 새로 출력되어 다시 1씩 증가시켰습니다.

〈결과〉



ChildView::OnPaint함수 속 코드입니다.

```
void CChildView::OnPaint()
{
    CPaintDC dc(this);

    CRect rect;
    GetClientRect(&rect); // 클라이언트 영역의 크기를 rect에 저장합니다.

    dc.MoveTo(rect.Width() / 2, 0); // 수직선을 그립니다.
    dc.LineTo(rect.Width() / 2, rect.Height());
    dc.MoveTo(0, rect.Height() / 2); // 수평선을 그립니다.
    dc.LineTo(rect.Width(), rect.Height() / 2);

    double y;

    for (int x = -rect.Width() / 2; x < rect.Width() / 2; x++)
    { // 중심부터 -끝부터 +끝까지 값을 증가시킵니다.
        y = sin(x * 3.14 / 180); // 라디안으로 변환 해 파라미터로 사용했습니다
        dc.SetPixelV(x + rect.Width() / 2, (int)(y * -100) + rect.Height() / 2, RGB(255, 0, 0));
    } // 삼각함수의 결과값은 double형이고 -1~1이기 때문에 100배 확대해 사용했습니다.

    // 코사인과 탄젠트 그래프입니다.
    for (int x = -rect.Width() / 2; x < rect.Width() / 2; x++)
    {
        y = cos(x * 3.14 / 180);
        dc.SetPixelV(x + rect.Width() / 2, (int)(y * -100) + rect.Height() / 2, RGB(0, 0, 255));
    }

    for (int x = -rect.Width() / 2; x < rect.Width() / 2; x++)
    {
        y = tan(x * 3.14 / 180);
        dc.SetPixelV(x + rect.Width() / 2, (int)(y * -100) + rect.Height() / 2, RGB(0, 0, 0));
    }

    // x좌표를 나타내는 부분입니다.
    dc.SetBkMode(TRANSPARENT);
    dc.SetTextAlign(TA_CENTER);

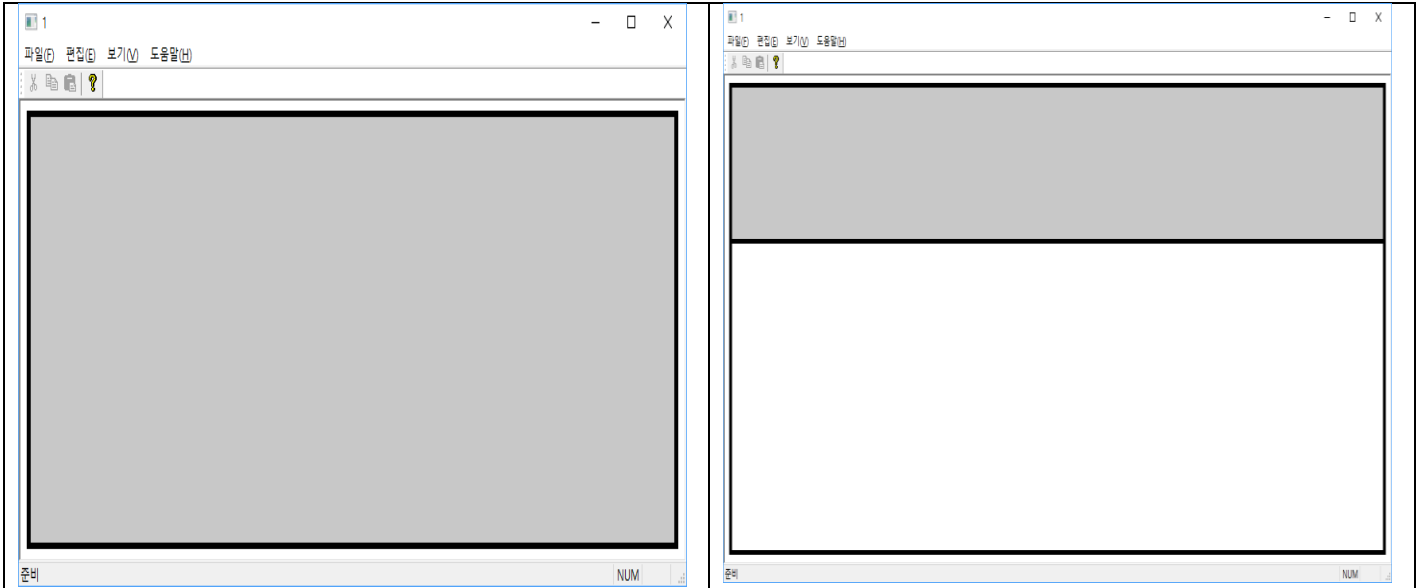
    int i = 0, k = 0;
    CString str[2];

    // 정중앙을 기준으로 +-90씩 출력했습니다.
    while (rect.Width() > rect.Width() / 2 + i)
    {
        str[0].Format(_T("%d"), i);
        dc.TextOutW(rect.Width() / 2 + i, rect.Height() / 2, str[0]);
        i += 90;

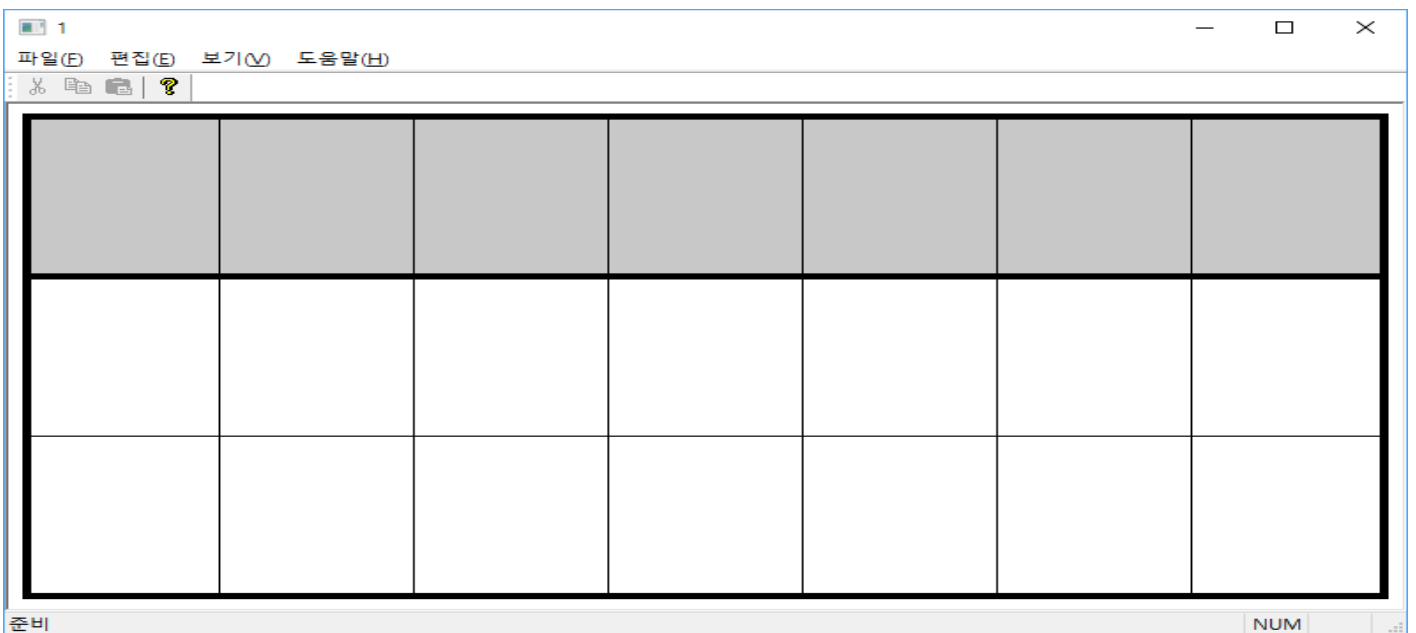
        str[1].Format(_T("%d"), k);
        dc.TextOutW(rect.Width() / 2 + k, rect.Height() / 2, str[1]);
        k -= 90;
    }
}
```

심화문제 1

문제로 나오는 표의 크기가 따로 없어 클라이언트의 크기에 따라 표와 글씨도 변하게 만들었습니다.
우선 두꺼운 선으로 되어있는 회색 사각형을 그리고 2/3 면적만큼 하얀 사각형을 다시 그리는 방법으로 1행의 회색과 2,3행의 흰색 바탕을 만들었습니다.



그 후 표의 너비와 높이를 등분해 가로줄과 세로줄을 그려 칸을 나누었습니다.



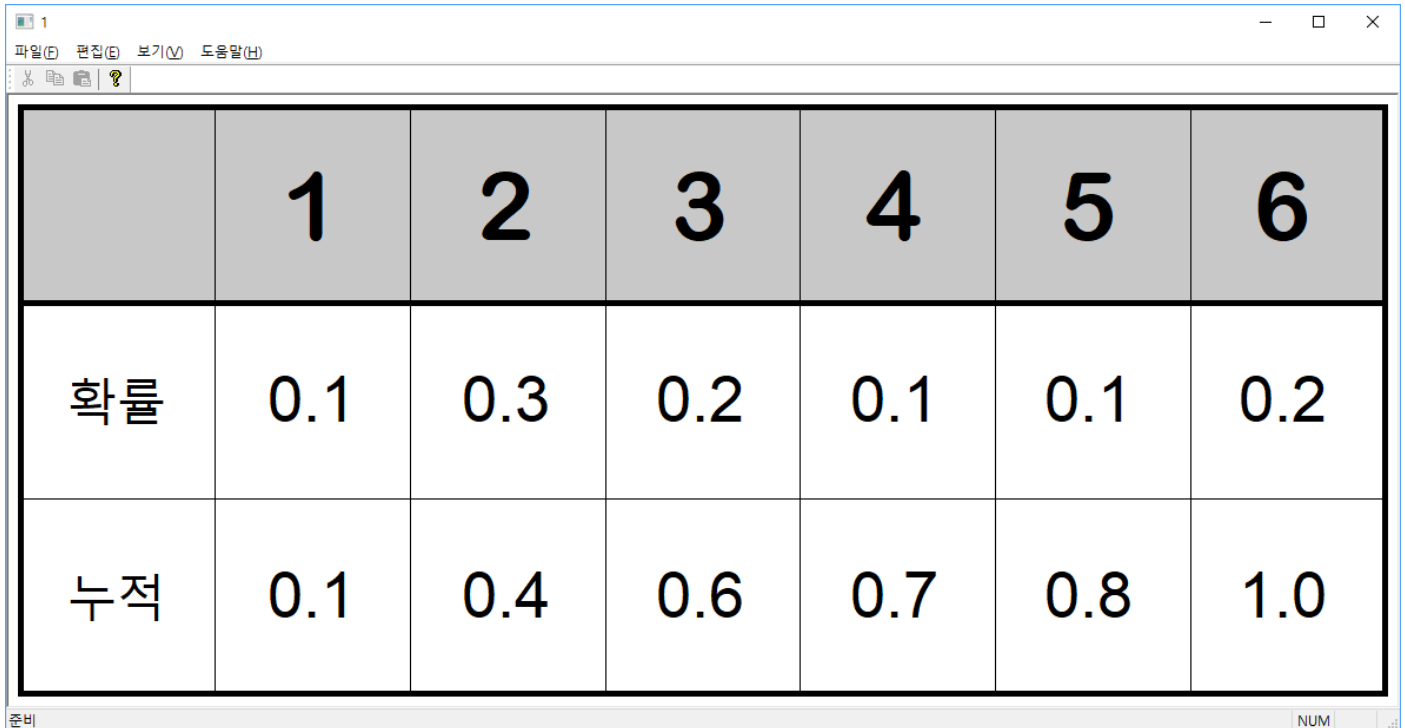
글자의 배경색을 투명하게 하고 1행은 숫자를 2, 3행은 미리 초기화한 CString 배열을 사용해 출력했습니다.

```
CString a[] = { _T("확률"), _T("0.1"), _T("0.3"), _T("0.2"), _T("0.1"), _T("0.1"), _T("0.2") };  
CString b[] = { _T("누적"), _T("0.1"), _T("0.4"), _T("0.6"), _T("0.7"), _T("0.8"), _T("1.0") };
```

CRect클래스를 이용해 각 칸만큼 사각형을 만들고 그 사각형의 중심에 글씨를 출력했습니다
다만 이미 위에서 각 칸들을 그렸기 때문에 따로 사각형을 출력하지는 않았습니다.

1행의 인덱스 폰트는 "Arial Rounded MT Bold"이며 2, 3행의 폰트는 "Arial" 입니다.

<결과>



	1	2	3	4	5	6
확률	0.1	0.3	0.2	0.1	0.1	0.2
누적	0.1	0.4	0.6	0.7	0.8	1.0

ChildView::OnPaint함수 속 코드입니다.

```
void CChildView::OnPaint()
{
    CPaintDC dc(this);
    CRect rect;
    GetClientRect(&rect);

    int w = rect.Width(); // 화면의 너비 w와
    int h = rect.Height(); // 화면의 높이 h 입니다.

    CBrush brush1(RGB(200, 200, 200)); // 회색으로 바탕색을 정하고
    dc.SelectObject(brush1);

    CPen pen1(PS_SOLID, 5, RGB(0, 0, 0));
    dc.SelectObject(&pen1); // 두꺼운 선을 설정하고

    dc.Rectangle(10, 10, w - 10, h - 10); // 배경색이 회색인 상하좌우 여백이 10인 사각형을 그립니다.

    CBrush brush2(RGB(255, 255, 255));
    dc.SelectObject(brush2); // 다시 흰색을 바탕색으로 설정하고

    dc.Rectangle(10, 10 + (h - 20) / 3, w - 10, h - 10);
    // 회색인 1행만 제외하고 나머지 두 행 부분을 다시 덧칠합니다.

    CPen pen2(PS_SOLID, 1, RGB(0, 0, 0));
    dc.SelectObject(&pen2); // 얇은 선

    dc.MoveTo(10, 10 + (h - 20) * 2 / 3);
    dc.LineTo(w - 10, 10 + (h - 20) * 2 / 3); // 얇은 선으로 그리는 가로 줄과

    for (int i = 1; i <= 7; i++)
    // 세로줄 7개를 그립니다. 한 칸은 너비 (w-20) / 7, 높이 (h - 20) / 3 입니다.
    {
        dc.MoveTo(10 + (w - 20) * i / 7, 10);
```

```

        dc.LineTo(10 + (w - 20) * i / 7, h - 10);
    }

    CFont font1;
    font1.CreatePointFont(w*h / 1000, _T("Arial Rounded MT Bold"));
    dc.SelectObject(font1);
    // Arial 볼드체로 설정합니다. 크기는 화면을 조절해도 같은 비율로 커지고 줄어들게 설정했습니다.

    dc.SetBkMode(TRANSPARENT); // 회색이 가려지지 않게 글자 배경을 투명하게 설정합니다.

    CString num;

    for (int i = 0; i < 7; i++)
    {
        num.Format(_T("%d"), i);

        CRect textRect(10 + (w - 20) / 7 * i, 10, 10 + (w - 20) / 7 * (i + 1), 10 + (h - 20) / 3);

        if (i != 0) // 0을 제외하고 1부터 6까지 까지 칸 정중앙에 표시합니다.
        //칸은 이미 위에서 그렸으므로 따로 사각형을 출력하진 않았습니다.
        {
            dc.DrawTextW(num, textRect,
                DT_CENTER | DT_VCENTER | DT_SINGLELINE);
        }
    }

    CString a[] = { _T("확률"), _T("0.1"), _T("0.3"), _T("0.2"), _T("0.1"), _T("0.1"), _T("0.2") };
    // 2행의 데이터입니다.
    CString b[] = { _T("누적"), _T("0.1"), _T("0.4"), _T("0.6"), _T("0.7"), _T("0.8"), _T("1.0") };
    // 3행의 데이터입니다.

    CFont font2;
    font2.CreatePointFont(w*h / 1500, _T("Arial"));
    // 2행과 3행은 볼드체가 아니기에 다시 폰트를 지정했습니다.
    dc.SelectObject(font2);

    //1행과 같은 방법으로 2행과 3행을 출력합니다.
    //역시 칸은 이미 그렸으므로 그리지 않고 데이터만 정렬해 출력합니다.
    for (int i = 0; i < 7; i++)
    {
        CRect textRect( 10+(w-20)/7*i , 10+(h-20)/3 , 10+(w-20)/7*(i+1) , 10+(h-20)*2/3 );

        dc.DrawTextW(a[i], textRect,
            DT_CENTER | DT_VCENTER | DT_SINGLELINE);
    }

    for (int i = 0; i < 7; i++)
    {
        CRect textRect( 10+(w-20)/7*i , 10+(h-20)*2/3 , 10+(w-20)/7*(i+1) , h-10 );

        dc.DrawTextW(b[i], textRect,
            DT_CENTER | DT_VCENTER | DT_SINGLELINE);
    }
}

```