EasyX 基础

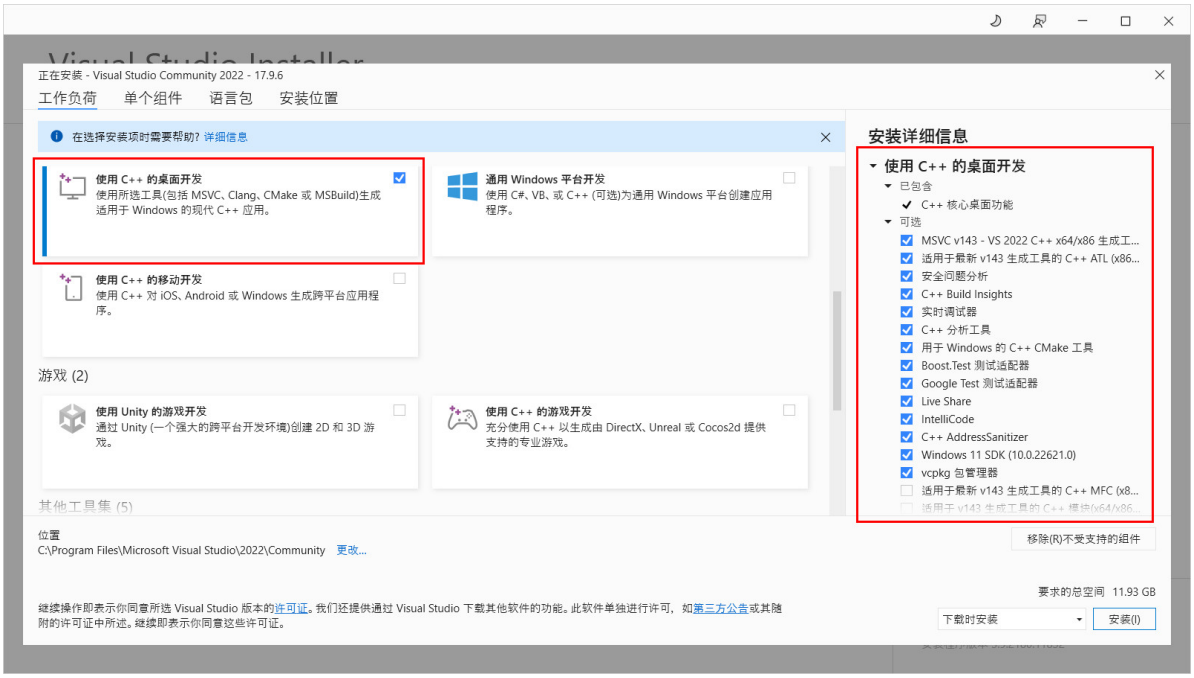
搭建开发环境

安装 Vuisual Studio

本次课程可以使用 VC++ 6.0、VS2005 ~ VS2022 任意版本，推荐使用最新版 **VS 2022**

官网地址：https://visualstudio.microsoft.com/zh-hans/downloads/ ，下载免费的**社区版**

双击安装文件 **VisualStudioSetup.exe**，只选择 “**使用C++的桌面开发**”，点击“**安装**”按钮进行在线安装



VS2022 常用快捷键：

代码排版：Ctrl + K, D

注释/取消注释：Ctrl + Shift + /

代码折叠：Ctrl + M, O

代码展开：Ctrl + M, P

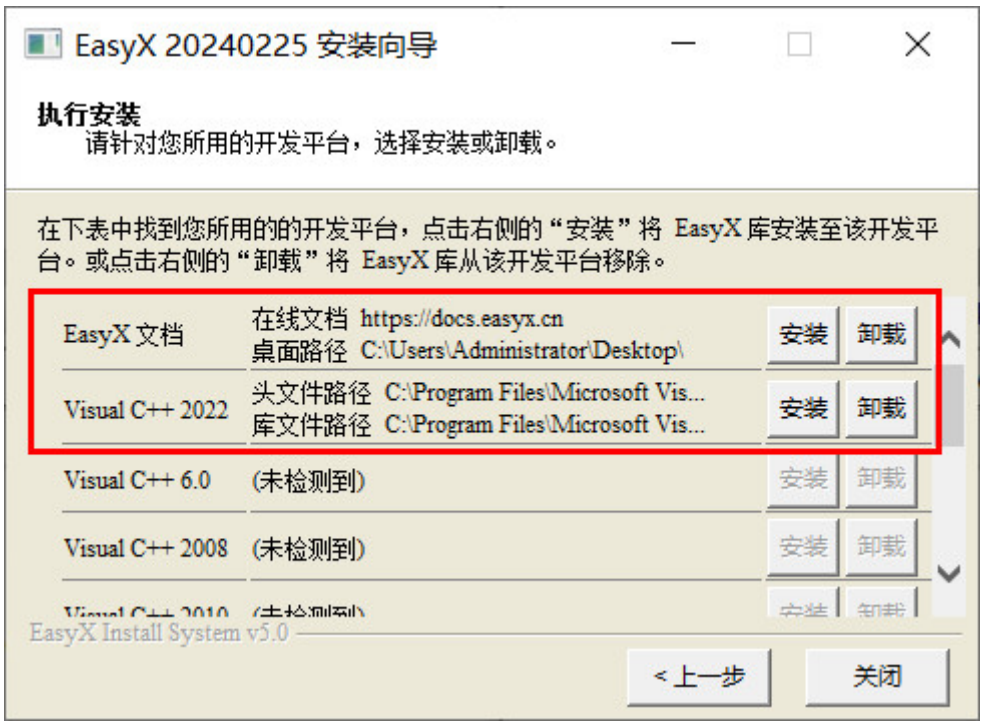
光标跳转到前一个位置：Ctrl + -

光标跳转到后一个位置：Ctrl + Shift + -

安装 EasyX 绘图库

EasyX Graphics Library 是针对 Visual C++ 的免费绘图库，支持 VC6.0 ~ VC2022，简单易用，学习成本极低，应用领域广泛。

EasyX 官网：https://easyx.cn/ ，下载最新版安装文件，双击 **EasyX_20240225.exe**



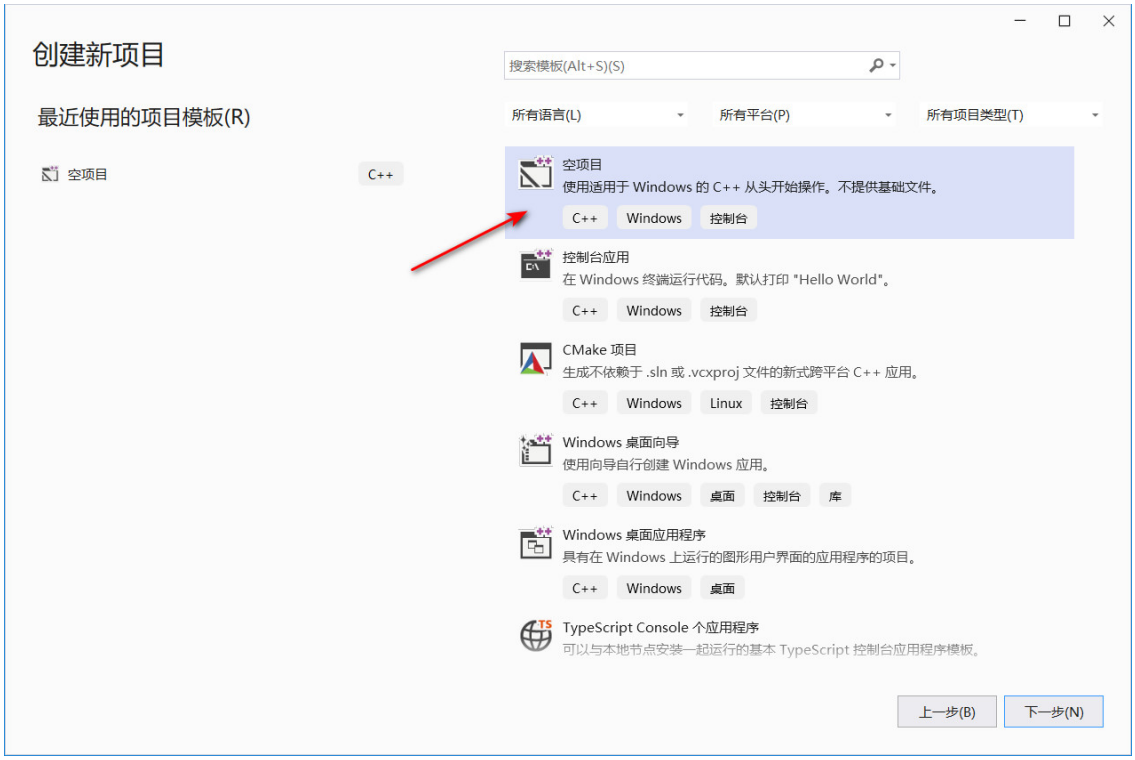
注1：EasyX 只支持 Visual Studio，在安装时会自动识别已安装的 VS 版本，**安装后需重启 VS 生效**

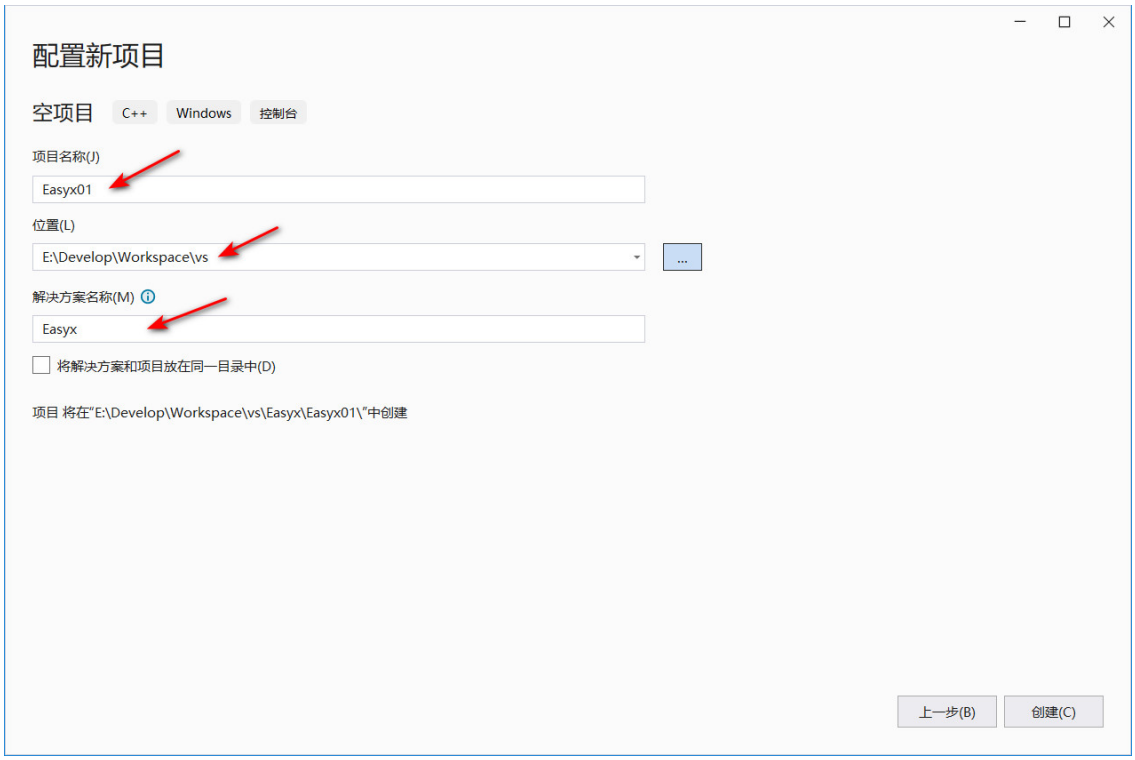
注2：EasyX 提供**在线**帮助文档 https://docs.easyx.cn/zh-cn/intro 和**离线**帮助文档 EasyX_Help.chm

注3：EasyX 向 VS 的 include 目录中写入 **graphics.h**（旧）、**easyx.h**（新） 两个头文件，向 lib 目录写入 EasyXa.lib、EasyXw.lib 两个静态库文件

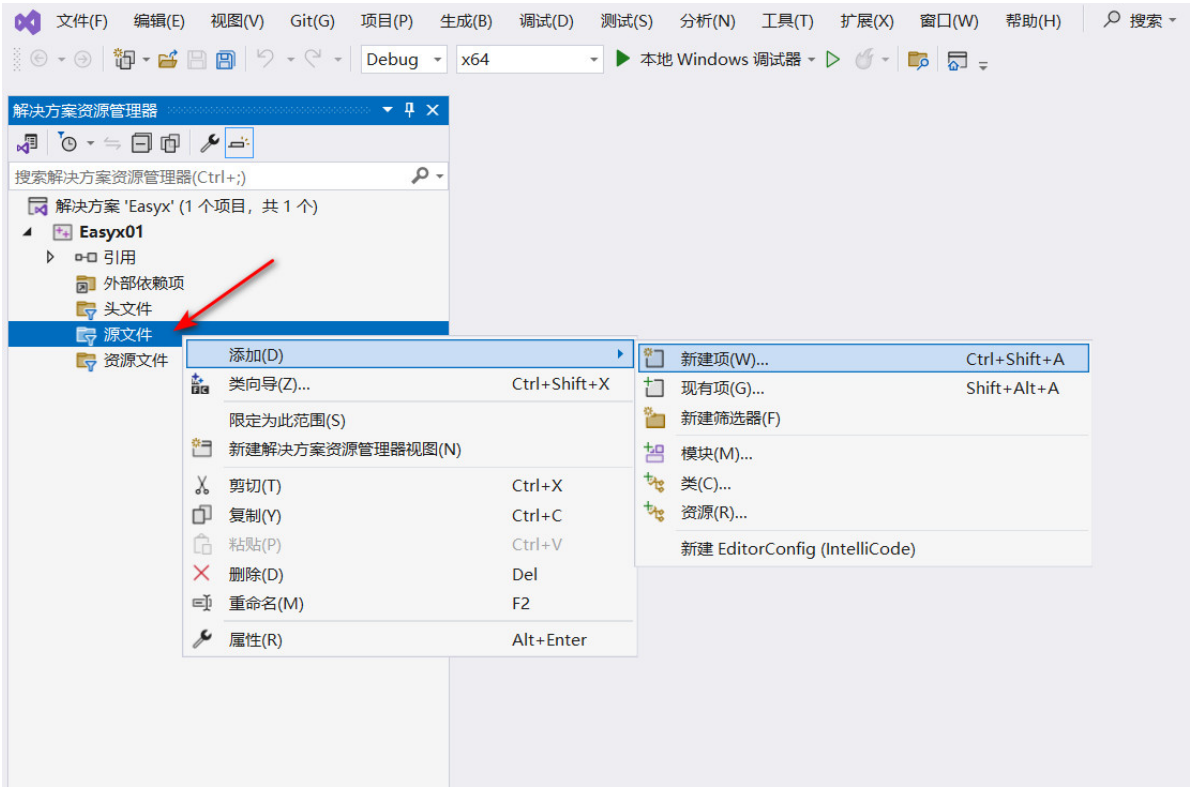
第一个 EasyX 程序

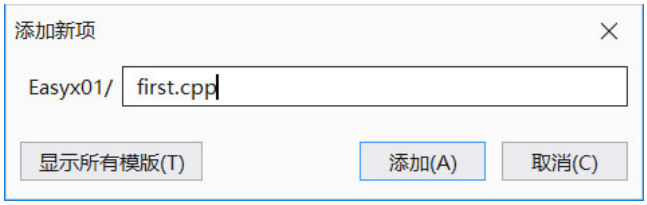
1. 启动 Visual Studio，创建一个 C++ 空项目





2. 在项目中创建一个.cpp源文件（右键源文件 -> 添加 -> 新建项 -> 设置文件名 first.cpp）





3. Easyx 范例

```
#include <graphics.h>        // 引用图形库头文件

int main()
{
    initgraph(800, 600);     // 创建绘图窗口，宽度800像素，高度600像素
    circle(200, 200, 100);   // 以(200, 200)坐标点为圆心，绘制半径为100像素的圆形
    system("pause");         // 按任意键继续
    closegraph();            // 关闭绘图窗口
    return 0;                // 成功退出
}
```

EasyX 基本概念

绘图窗口与设备

initgraph 函数用于初始化绘图窗口

```
HWND initgraph(
    int width,
    int height,
    int flag = NULL
);
```

示例1：创建禁用最小化和关闭按钮的绘图窗口

```
initgraph(800, 600, EX_NOMINIMIZE | EX_NOCLOSE);
```

示例2：窗口开启 EX_SHOWCONSOLE 模式，可以进行代码调试

```
initgraph(800, 600, EX_SHOWCONSOLE);
```

在 EasyX 中，**设备**分两种，一种是默认的**绘图窗口**，另一种是 **IMAGE 对象。**

通过 SetWorkingImage 函数可以设置当前用于绘图的设备。设置当前用于绘图的设备后，所有的绘图函数都会绘制在该设备上。

坐标

在 EasyX 中，坐标分两种：物理坐标和逻辑坐标。

- **物理坐标**

物理坐标是描述设备的坐标体系。

坐标原点在设备的左上角，X 轴向右为正，Y 轴向下为正，度量单位是像素（Pixel）。

坐标原点、坐标轴方向、缩放比例都不能改变。

- **逻辑坐标**

逻辑坐标是在程序中用于绘图的坐标体系。

坐标默认的原点在窗口的左上角，X 轴向右为正，Y 轴向下为正，度量单位是点。

默认情况下，逻辑坐标与物理坐标是一一对应的，一个逻辑点等于一个物理像素。

在本手册中，凡是没有注明的坐标，均指逻辑坐标。

坐标相关函数

函数用法	函数说明
void **setorigin** (int x, int y)	用于设置坐标原点。

函数用法	函数说明
void **setaspectratio** (float xasp, float yasp)	通过设置 x 和 y 方向上的缩放因子，从而修改绘图的缩放比例或坐标轴方向。

范例：

```
#include <graphics.h>

int main()
{
    initgraph(600, 600);

    setorigin(300, 300);        // 将绘图窗口的中心点作为坐标原点
    circle(0, 0, 100);

    setorigin(0, 0);            // 将绘图窗口的左上角作为坐标原点
    setaspectratio(2, 1);       // x轴方向的缩放因子为2，y轴方向的缩放因子为1(默认值)
    circle(100, 100, 100);

    setorigin(0, 600);          // 将绘图窗口的左下角作为坐标原点
    setaspectratio(1, -1);      // 缩放因子为负数，可以实现坐标轴的翻转，此行可使y轴向上为
正
    circle(100, 100, 100);

    system("pause");
    closegraph();
    return 0;
}
```

颜色

EasyX 使用 24bit 真彩色，有四种表示颜色的方法：https://docs.easyx.cn/zh-cn/color，通过 **setlinecolor** 函数可以设置线条颜色

1. 用预定义常量表示颜色 （ 常量名要大写 ）
2. 用 RGB 宏合成颜色 （ RGB(RRGGBB) ）
3. 用16进制数字表示颜色 （ 0xBBGGRR ），注意颜色的顺序与RGB宏相反
4. 用 HSLtoRGB、HSVtoRGB 转换其他色彩模型到 RGB 颜色

范例：

```
#include <graphics.h>

int main()
{
    initgraph(800, 600);

    setfillcolor(BLUE);                     // 用预定义常量表示颜色
```

```
    solidcircle(100, 200, 100);

    setfillcolor(RGB(0, 0, 170));           // 用RGB宏合成颜色
    solidcircle(300, 200, 100);

    setfillcolor(0xaa0000);                 // 用16进制数字表示颜色
    solidcircle(500, 200, 100);

    setfillcolor(HSLtoRGB(240, 1, 0.33));   // 用 HSLtoRGB、HSVtoRGB 转换其他色彩模型到 RGB 颜色
    solidcircle(700, 200, 100);

    system("pause");
    closegraph();
    return 0;
}
```

EasyX 图形绘制函数（33个）

https://docs.easyx.cn/zh-cn/drawing-func

函数用法	函数说明
void **circle** (int x, int y, int radius)	画无填充的圆
fillcircle	画有边框的填充圆
solidcircle	画无边框的填充圆
clearcircle	用当前背景色清空圆形区域
void **ellipse** (int left, int top, int right, int bottom)	画无填充的椭圆
fillellipse	画有边框的填充椭圆
solidellipse	画无边框的填充椭圆
clearellipse	用当前背景色清空椭圆区域
void **pie** (int left, int top, int right, int bottom, double stangle, double endangle);	画无填充的扇形
fillpie	画有边框的填充扇形
solidpie	画无边框的填充扇形

函数用法	函数说明
clearpie	用当前背景色清空扇形区域
void **rectangle** (int left, int top, int right, int bottom)	画无填充的矩形
fillrectangle	画有边框的填充矩形
solidrectangle	画无边框的填充矩形
clearrectangle	用当前背景色清空矩形区域
void **roundrect** (int left, int top, int right, int bottom, int ellipsewidth, int ellipseheight)	画无填充的圆角矩形
fillroundrect	画有边框的填充圆角矩形
solidroundrect	画无边框的填充圆角矩形
clearroundrect	用当前背景色清空圆角矩形区域
void **polygon** (const POINT *points, int num);	画无填充的多边形
fillpolygon	画有边框的填充多边形
solidpolygon	画无边框的填充多边形
clearpolygon	用当前背景色清空多边形区域
void **putpixel** (int x, int y, COLORREF color)	画点
void **line** (int x1, int y1, int x2, int y2)	画直线
void **arc** (int left, int top, int right, int bottom, double stangle, double endangle)	画椭圆弧
void **polyline** (const POINT *points, int num)	画多条连续的直线
void **polybezier** (const POINT *points, int num)	画三次方贝塞尔曲线
void **floodfill** (int x, int y, COLORREF color, int filltype = FLOODFILLBORDER)	填充区域

函数用法	函数说明
COLORREF **getpixel** (int x, int y)	获取坐标点的颜色
int **getwidth** ( )	获取绘图区的宽度
int **getheight** ( )	获取绘图区的高度

范例

范例：矩形

```
#include <graphics.h>

int main()
{
    initgraph(600, 600);

    rectangle(100, 100, 300, 200);      // 有边框无填充的矩形

    setlinecolor(YELLOW);
    setfillcolor(RED);
    fillrectangle(350, 100, 550, 200);  // 有边框的填充矩形

    setfillcolor(GREEN);
    solidrectangle(100, 400, 300, 500); // 无边框的填充矩形

    setbkcolor(BLUE);
    clearrectangle(350, 400, 550, 500); // 用背景色填充矩形区域

    system("pause");
    closegraph();
    return 0;
}
```

范例：扇形

```
#include <graphics.h>

#define PI 3.14

int main()
{
    initgraph(600, 600);
    setlinecolor(YELLOW);

    // rectangle(100, 100, 300, 300);
    pie(100, 100, 300, 300, 0, PI/2);
```

```
    // rectangle(350, 100, 550, 300);
    fillpie(350, 100, 550, 300, PI/4, 3*PI/4);

    // rectangle(100, 350, 300, 550);
    solidpie(100, 350, 300, 550, PI/2, 2*PI);

    system("pause");
    closegraph();
    return 0;
}
```

范例：圆角矩形

```
#include <graphics.h>


int main()
{
    initgraph(600, 600);

    setlinecolor(YELLOW);
    roundrect(200, 100, 400, 200, 50, 50);          // 无填充的圆角矩形

    setfillcolor(RED);
    fillroundrect(200, 250, 400, 350, 50, 50);      // 有边框的填充圆角矩形

    solidroundrect(200, 400, 400, 500, 50, 50);     // 无边框的填充圆角矩形

    system("pause");
    closegraph();
    return 0;
}
```

范例：多边形

```
#include <graphics.h>


int main()
{
    initgraph(600, 600);
    setlinecolor(YELLOW);

    // 绘制三角形：方法1
    POINT pts1[] = { {50, 200}, {200, 200}, {200, 50} };
    polygon(pts1, 3);

    // 绘制三角形：方法2
    int pts2[] = { 300, 200,  450, 200,  450, 50 };
    polygon((POINT*)pts2, 3);
```

```
    system("pause");
    closegraph();
    return 0;
}
```

范例：贝塞尔曲线

```
#include <graphics.h>

int main()
{
    initgraph(600, 600);

    //               起始点      控制点1         控制点2         终点/起点    控制点1
控制点2         终点
    POINT pts[] = { {150, 200}, {160, 150}, {240, 150}, {250, 100}, {260, 150},
{340, 150}, {350, 200} };

    solidcircle(150, 200, 3);   // 起始点
    solidcircle(160, 150, 3);   // 控制点1
    solidcircle(240, 150, 3);   // 控制点2
    solidcircle(250, 100, 3);   // 终点/起点
    solidcircle(260, 150, 3);   // 控制点1
    solidcircle(340, 150, 3);   // 控制点2
    solidcircle(350, 200, 3);   // 终点

    setlinecolor(DARKGRAY);
    polyline(pts, 7);       // 画灰色的辅助线

    setlinecolor(GREEN);
    polybezier(pts, 7);     // 画绿色的贝塞尔曲线

    system("pause");
    closegraph();
    return 0;
}
```

范例：五角星

```
#include <graphics.h>
#include <math.h>
#define PI 3.14159

int main()
{
    initgraph(600, 600);

    setorigin(300, 300);            // 设置原点在屏幕中间
    setaspectratio(1, -1);          // 转换y坐标方向
    setbkcolor(RED);                // 设置背景色为红色
    cleardevice();                  // 用当前背景色清除设备
```

```
    setlinecolor(YELLOW);           // 设置画线颜色
    setlinestyle(PS_SOLID, 3);      // 设置画线样式
    int r1 = 200;                   // 外接圆的半径
    int r2 = 80;                    // 内接圆的半径
    int g = 18;                     // 第一个点初始角度
    int x1[5], y1[5];               // 外接圆五个点的坐标
    int x2[5], y2[5];               // 内接圆五个点的坐标
    for (int i = 0; i < 5; i++) {   // 计算10个点的坐标
        x1[i] = r1 * cos(g * PI / 180);
        y1[i] = r1 * sin(g * PI / 180);
        x2[i] = r2 * cos((g + 36) * PI / 180);
        y2[i] = r2 * sin((g + 36) * PI / 180);
        line(x1[i], y1[i], x2[i], y2[i]);   // 先画五条线（外径点到内径点）
        g += 72;                            // 变换角度
    }

    // 再画五条线（内径点到外径点）
    //line(x2[0], y2[0], x1[1], y1[1]);
    //line(x2[1], y2[1], x1[2], y1[2]);
    //line(x2[2], y2[2], x1[3], y1[3]);
    //line(x2[3], y2[3], x1[4], y1[4]);
    //line(x2[4], y2[4], x1[0], y1[0]);
    for (int i = 0; i < 5; i++)
        line(x2[i], y2[i], x1[(i + 1) % 5], y1[(i + 1) % 5]);

    setfillcolor(YELLOW);               // 设置填充颜色
    floodfill(0, 0, getlinecolor());    // 从封闭图形内的指定坐标开始，将封闭区间用黄色
填充（直到画线边界）

    system("pause");
    closegraph();
    return 0;
}
```

范例：五角星分形图

```
#include <graphics.h>
#include <math.h>
#define PI 3.14159

int main()
{
    initgraph(600, 600);

    setlinecolor(YELLOW);
    setorigin(300, 300);
    setaspectratio(1, -1);
    setlinestyle(PS_SOLID, 3);
    int x1[5], y1[5];
    int x2[5], y2[5];
    int r1 = 200;
    int r2 = 50;
    for (int t = 0; t < 360; t += 20) {             // 旋转绘图360/20=18次
```

```
        int g = t + 18;                                    // 第一个点初始角度
        for (int i = 0; i < 5; i++) {
            x1[i] = r1 * cos(g * PI / 180);
            y1[i] = r1 * sin(g * PI / 180);
            x2[i] = r2 * cos((g + 36) * PI / 180);
            y2[i] = r2 * sin((g + 36) * PI / 180);
            line(x1[i], y1[i], x2[i], y2[i]);
            g += 72;
        }
        for (int i = 0; i < 5; i++)
            line(x2[i], y2[i], x1[(i + 1) % 5], y1[(i + 1) % 5]);
    }

    system("pause");
    closegraph();
    return 0;
}
```

EasyX 文字输出函数（8个）

https://docs.easyx.cn/zh-cn/text-func

函数用法	函数说明
void **outtextxy** (int x, int y, LPCTSTR str) void **outtextxy** (int x, int y, TCHAR c)	在指定位置输出字符或字符串
int **drawtext** (LPCTSTR str, RECT* pRect, UINT uFormat) int **drawtext** (TCHAR c, RECT* pRect, UINT uFormat)	在指定区域内以指定格式输出字符或字符串
void **settextcolor** (COLORREF color)	设置当前文字颜色
COLORREF **gettextcolor** ()	获取当前文字颜色
void **settextstyle** (int nHeight, int nWidth, LPCTSTR lpszFace) void **settextstyle** (const LOGFONT *font)	设置当前文字样式（宽、高、字体）
void **gettextstyle** (LOGFONT *font)	获取当前文字样式
LOGFONT	文字样式的结构体
int **textheight** (LPCTSTR str) int **textheight** (TCHAR c)	获取字符串实际占用的像素高度

函数用法	函数说明
int **textwidth** (LPCTSTR str) int **textwidth** (TCHAR c)	获取字符串实际占用的像素宽度

范例

范例：outtextxy 输出字符串

```
#include <graphics.h>

int main()
{
    initgraph(600, 600);

    LPCTSTR text = _T("EasyX 绘图");              // 待输出的文字
    settextcolor(YELLOW);                         // 设置文字颜色
    settextstyle(60, 30, _T("宋体"));             // 设置字高、字宽、字体

    outtextxy(0, 0, text);                        // 在屏幕左上角绘制文字
    int h = textheight(text);                     // 获取字高
    int w = textwidth(text);                      // 获取字宽
    outtextxy(300 - w/2, 300 - h/2, text);        // 在屏幕中心绘制文字

    system("pause");
    closegraph();
    return 0;
}
```

范例：outtextxy 输出字符串（抗锯齿效果）

```
#include <graphics.h>

int main()
{
    initgraph(600, 600);

    LPCTSTR text = _T("EasyX 绘图");              // 待输出的文字
    settextcolor(YELLOW);                         // 设置文字颜色

    LOGFONT f;                                    // 定义LOGFONT结构体变量
    gettextstyle(&f);                             // 获取当前字体设置
    f.lfHeight = 60;                              // 设置字高为60
    _tcscpy_s(f.lfFaceName, _T("宋体"));          // 设置字体为黑体
    f.lfQuality = ANTIALIASED_QUALITY;            // 设置输出效果为抗锯齿
    settextstyle(&f);                             // 设置字体样式

    int h = textheight(text);                     // 获取字高
    int w = textwidth(text);                      // 获取字宽
    outtextxy(300 - w / 2, 300 - h / 2, text);    // 在屏幕中心绘制文字
```

```
    system("pause");
    closegraph();
    return 0;
}
```

范例：drawtext 输出字符串

```
#include <graphics.h>
#define WIN_WIDTH 1000
#define WIN_HEIGHT 600

int main()
{
    initgraph(WIN_WIDTH, WIN_HEIGHT);

    settextcolor(YELLOW);
    settextstyle(60, 30, _T("宋体"));

    // 在屏幕中心绘制文字
    RECT r = { 0, 0, WIN_WIDTH-1, WIN_HEIGHT-1 };
    drawtext(_T("EasyX 绘图"), &r, DT_CENTER | DT_VCENTER | DT_SINGLELINE);

    system("pause");
    closegraph();
    return 0;
}
```

EasyX 综合范例

范例1：字符阵

```
#include <graphics.h>
#include <time.h>
#include <conio.h>

int main()
{
    // 设置随机种子
    srand((unsigned) time(NULL));

    // 初始化图形模式
    initgraph(640, 480);

    int  x, y;
    char c;

    settextstyle(16, 8, _T("Courier")); // 设置字体
```

```
    // 设置颜色
    settextcolor(GREEN);
    setlinecolor(BLACK);

    for (int i = 0; i <= 479; i++)
    {
        // 在随机位置显示三个随机字母
        for (int j = 0; j < 3; j++)
        {
            x = (rand() % 80) * 8;
            y = (rand() % 20) * 24;
            c = (rand() % 26) + 65;
            outtextxy(x, y, c);
        }

        // 画线擦掉一个像素行
        line(0, i, 639, i);

        Sleep(10);                  // 延时
        if (i >= 479)   i = -1;
        if (_kbhit())   break;      // 按任意键退出
    }

    // 关闭图形模式
    closegraph();
    return 0;
}
```

范例2：星空

```
#include <graphics.h>
#include <time.h>
#include <conio.h>

#define MAXSTAR 200 // 星星总数

struct STAR
{
    double  x;
    int     y;
    double  step;
    int     color;
};

STAR star[MAXSTAR];

// 初始化星星
void InitStar(int i)
{
    star[i].x = 0;
    star[i].y = rand() % 480;
    star[i].step = (rand() % 5000) / 1000.0 + 1;
    star[i].color = (int)(star[i].step * 255 / 6.0 + 0.5);  // 速度越快，颜色越亮
```

```
    star[i].color = RGB(star[i].color, star[i].color, star[i].color);
}

// 移动星星
void MoveStar(int i)
{
    // 擦掉原来的星星
    putpixel((int)star[i].x, star[i].y, 0);

    // 计算新位置
    star[i].x += star[i].step;
    if (star[i].x > 640)    InitStar(i);

    // 画新星星
    putpixel((int)star[i].x, star[i].y, star[i].color);
}

// 主函数
int main()
{
    srand((unsigned)time(NULL));    // 随机种子
    initgraph(640, 480);            // 创建绘图窗口

    // 初始化所有星星
    for(int i = 0; i < MAXSTAR; i++)
    {
        InitStar(i);
        star[i].x = rand() % 640;
    }

    // 绘制星空，按任意键退出
    while(!_kbhit())
    {
        for(int i = 0; i < MAXSTAR; i++)
            MoveStar(i);
        Sleep(20);
    }

    closegraph();                   // 关闭绘图窗口
    return 0;
}
```

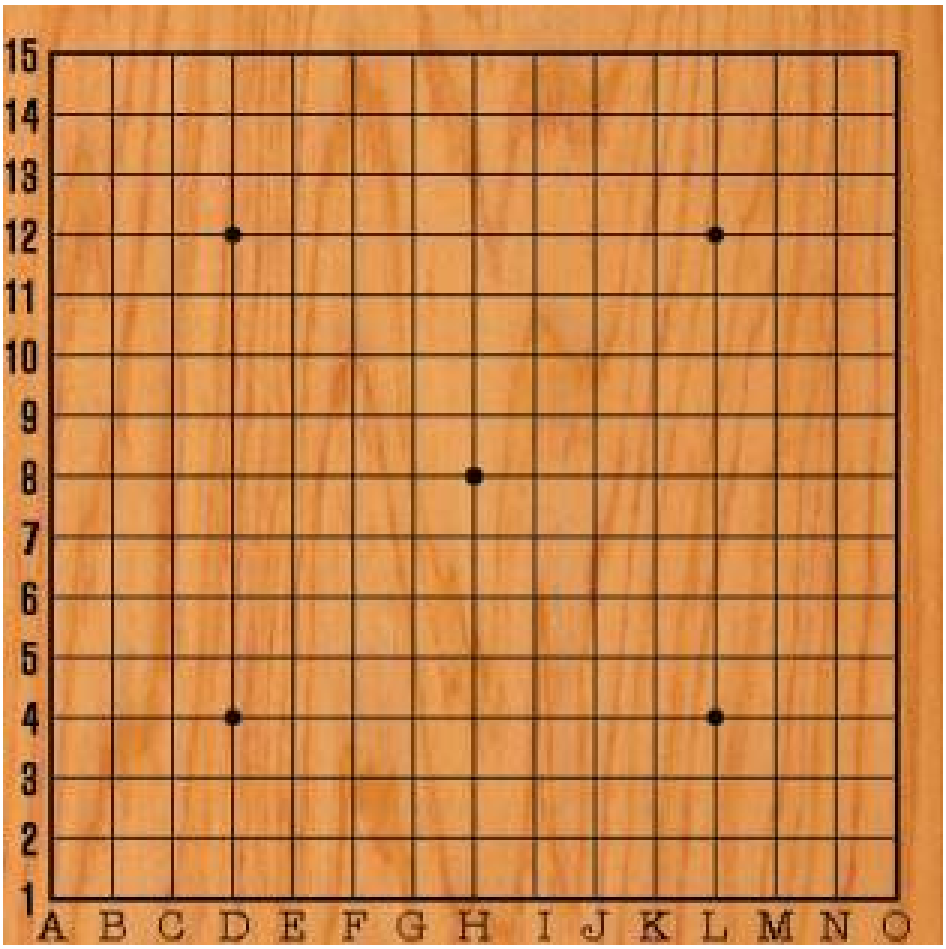
范例3：五星红旗

```
#include<graphics.h>
#include<math.h>
#define PI 3.14159

/*
    drawstar函数参数说明
        x,y:        五角星中心点坐标
        outter_r:   外接圆半径
        inner_r:    内接圆半径
```

```
        initangle: 第一个点初始角度
*/
void drawstar(int x, int y, int outter_r, int inner_r, int initangle) {
    setlinecolor(YELLOW);               // 设置画线颜色
    setorigin(x, y);                    // 设置原点在屏幕中间
    setaspectratio(1, -1);              // 转换y坐标方向
    setlinestyle(PS_SOLID, 3);          // 设置画线样式
    int r1 = outter_r;                  // 外接圆的半径
    int r2 = inner_r;                   // 内接圆的半径
    int g = initangle;                  // 第一个点初始角度
    int x1[5], y1[5];                   // 外接圆五个点的坐标
    int x2[5], y2[5];                   // 内接圆五个点的坐标
    for (int i = 0; i < 5; i++) {       // 计算10个点的坐标
        x1[i] = r1 * cos(g * PI / 180);
        y1[i] = r1 * sin(g * PI / 180);
        x2[i] = r2 * cos((g + 36) * PI / 180);
        y2[i] = r2 * sin((g + 36) * PI / 180);
        line(x1[i], y1[i], x2[i], y2[i]);       // 先画五条线
        g += 72;
    }

    // 再画五条线
    for (int i = 0; i < 5; i++)
        line(x2[i], y2[i], x1[(i + 1) % 5], y1[(i + 1) % 5]);

    setfillcolor(YELLOW);
    floodfill(0, 0, getlinecolor());    // 将黄色边界内的空间用红色填满
}

int main()
{
    initgraph(900, 600);

    setbkcolor(RGB(255, 0, 0));         // 设置背景色为红色
    cleardevice();                      // 使用当前背景色清空绘图设备
    drawstar(150, 150, 100, 40, 18);    // 大五角星
    drawstar(300, 50, 25, 10, 72);      // 小五角星1
    drawstar(350, 120, 25, 10, 45);     // 小五角星2
    drawstar(350, 210, 25, 10, 18);     // 小五角星3
    drawstar(300, 280, 25, 10, 72);     // 小五角星4

    system("pause");
    closegraph();
    return 0;
}
```

随堂练习

练习1：绘制五子棋棋盘



练习2：绘制时钟

https://pic.sogou.com/pics?ie=utf8&p=40230504&interV=kKIOkrELjbkRmLkElbkTkKIMkrELjboImLkEk74TkKIRmLkEk78TkKILkY%3D%3D_-115092183&query=%E6%97%B6%E9%92%9F

范例：GetLocalTime 获取系统时间

```
#include <graphics.h>
#include <time.h>
#include <stdio.h>

int main()
{
    SYSTEMTIME time;                        // 定义系统时间变量
    int second;                             // 秒数
    int minute;                             // 分钟数
    int hour;                               // 小时数

    GetLocalTime(&time);                    // 获取本地系统时间
    second = time.wSecond;                  // 设置秒数
    minute = time.wMinute;                  // 设置分钟数
    hour = time.wHour;                      // 设置小时数
    printf("现在的时间是：%2d:%2d:%2d", hour, minute, second);

    return 0;
}
```