# 学习大纲

|  |
| --- |
|  |

## FillRect函数的用法

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **FillRect** 函数使用指定的画笔填充矩形。 此函数包括左边框和上边框，但不包括矩形的右边框和下边框。 语法 C++复制  int FillRect(  [in] HDC hDC,  [in] const RECT \*lprc,  [in] HBRUSH hbr  ); 参数 [in] hDC  设备上下文的句柄。  [in] lprc  指向 [RECT](https://learn.microsoft.com/zh-cn/windows/desktop/api/windef/ns-windef-rect) 结构的指针，该结构包含要填充的矩形的逻辑坐标。  [in] hbr  用于填充矩形的画笔的句柄。 返回值 如果该函数成功，则返回值为非零值。  如果函数失败，则返回值为零。 注解 *由 hbr* 参数标识的画笔可以是逻辑画笔的句柄，也可以是颜色值。 如果为逻辑画笔指定句柄，请调用以下函数之一以获取句柄： [CreateHatchBrush](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-createhatchbrush)、 [CreatePatternBrush](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-createpatternbrush) 或 [CreateSolidBrush](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-createsolidbrush)。 此外，还可以使用 [GetStockObject](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-getstockobject) 函数检索其中一个股票画笔的句柄。 如果为 *hbr* 参数指定颜色值，则它必须是标准系统颜色之一， (必须将值 1 添加到所选颜色) 。 例如：  C++  FillRect(hdc, &rect, (HBRUSH) (COLOR\_WINDOW+1));  有关所有标准系统颜色的列表，请参阅 [GetSysColor](https://learn.microsoft.com/zh-cn/windows/desktop/api/winuser/nf-winuser-getsyscolor)。  填充指定矩形时， **FillRect** 不包括矩形的右侧和底部。 无论当前映射模式如何，GDI 都会填充一个矩形，但不包括右列和底部行。 示例 有关示例，请参阅 [使用矩形](https://learn.microsoft.com/zh-cn/windows/desktop/gdi/using-rectangles)。 要求  |  |  | | --- | --- | | **标头** | winuser.h (包括 Windows.h) | | **Library** | User32.lib | | **DLL** | User32.dll | | **API 集** | 在 Windows 8) 中引入的 ext-ms-win-ntuser-gui-l1-1-0 ( |  另请参阅 [CreateHatchBrush](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-createhatchbrush)  [CreatePatternBrush](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-createpatternbrush)  [CreateSolidBrush](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-createsolidbrush)  [填充的形状函数](https://learn.microsoft.com/zh-cn/windows/desktop/gdi/filled-shape-functions)  [填充形状概述](https://learn.microsoft.com/zh-cn/windows/desktop/gdi/filled-shapes)  [GetStockObject](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-getstockobject)  [矩形](https://learn.microsoft.com/zh-cn/windows/desktop/api/windef/ns-windef-rect) |

## FrameRect函数的用法

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **FrameRect** 函数使用指定的画笔在指定矩形周围绘制边框。 边框的宽度和高度始终是一个逻辑单元。 语法 C++复制  int FrameRect(  [in] HDC hDC,  [in] const RECT \*lprc,  [in] HBRUSH hbr  ); 参数 [in] hDC  在其中绘制边框的设备上下文的句柄。  [in] lprc  指向 [RECT](https://learn.microsoft.com/zh-cn/windows/desktop/api/windef/ns-windef-rect) 结构的指针，该结构包含矩形左上角和右下角的逻辑坐标。  [in] hbr  用于绘制边框的画笔的句柄。 返回值 如果该函数成功，则返回值为非零值。  如果函数失败，则返回值为零。 注解 *由 hbr* 参数标识的画笔必须已使用 [CreateHatchBrush](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-createhatchbrush)、[CreatePatternBrush](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-createpatternbrush) 或 [CreateSolidBrush](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-createsolidbrush) 函数创建，或使用 [GetStockObject](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-getstockobject) 函数检索。  如果 [RECT](https://learn.microsoft.com/zh-cn/windows/desktop/api/windef/ns-windef-rect) 结构的**底部**成员小于**顶部**成员，或者**如果右**成员小于**左侧**成员，则函数不会绘制矩形。 要求  |  |  | | --- | --- | | **标头** | winuser.h (包括 Windows.h) | | **Library** | User32.lib | | **DLL** | User32.dll | | **API 集** | 在 Windows 8.1) 中引入的 ext-ms-win-ntuser-gui-l1-1-1 ( | |  |  |  另请参阅 [CreateHatchBrush](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-createhatchbrush)  [CreatePatternBrush](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-createpatternbrush)  [CreateSolidBrush](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-createsolidbrush)  [填充的形状函数](https://learn.microsoft.com/zh-cn/windows/desktop/gdi/filled-shape-functions)  [填充形状概述](https://learn.microsoft.com/zh-cn/windows/desktop/gdi/filled-shapes)  [GetStockObject](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-getstockobject)  [矩形](https://learn.microsoft.com/zh-cn/windows/desktop/api/windef/ns-windef-rect) |

## InvertRect函数的用法

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **InvertRect** 函数通过对矩形内部中每个像素的颜色值执行逻辑 NOT 运算来反转窗口中的矩形。 语法 C++复制  BOOL InvertRect(  [in] HDC hDC,  [in] const RECT \*lprc  ); 参数 [in] hDC  设备上下文的句柄。  [in] lprc  指向 [RECT](https://learn.microsoft.com/zh-cn/windows/desktop/api/windef/ns-windef-rect) 结构的指针，该结构包含要反转的矩形的逻辑坐标。 返回值 如果该函数成功，则返回值为非零值。  如果函数失败，则返回值为零。 注解 在单色屏幕上， **InvertRect** 将白色像素设置为黑色，将黑色像素设置为白色。 在彩色屏幕上，反转取决于为屏幕生成颜色的方式。 为同一矩形调用 **两次 InvertRect** 会将显示还原到以前的颜色。 要求  |  |  | | --- | --- | | **标头** | winuser.h (包括 Windows.h) | | **Library** | User32.lib | | **DLL** | User32.dll | | **API 集** | 在 Windows 8.1) 中引入的 ext-ms-win-ntuser-gui-l1-1-1 ( |  另请参阅 [FillRect](https://learn.microsoft.com/zh-cn/windows/desktop/api/winuser/nf-winuser-fillrect)  [填充形状函数](https://learn.microsoft.com/zh-cn/windows/desktop/gdi/filled-shape-functions)  [填充形状概述](https://learn.microsoft.com/zh-cn/windows/desktop/gdi/filled-shapes)  [矩形](https://learn.microsoft.com/zh-cn/windows/desktop/api/windef/ns-windef-rect) |

## SetRect函数的用法

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **SetRect** 函数设置指定矩形的坐标。 这相当于将左侧、顶部、右侧和底部参数分配给 **RECT** 结构的相应成员。 语法 C++  BOOL SetRect(  [out] LPRECT lprc,  [in] int xLeft,  [in] int yTop,  [in] int xRight,  [in] int yBottom  ); 参数 [out] lprc  指向包含要设置的矩形的 [RECT](https://learn.microsoft.com/zh-cn/windows/desktop/api/windef/ns-windef-rect) 结构的指针。  [in] xLeft  指定矩形左上角的 x 坐标。  [in] yTop  指定矩形左上角的 y 坐标。  [in] xRight  指定矩形右下角的 x 坐标。  [in] yBottom  指定矩形右下角的 y 坐标。 返回值 如果该函数成功，则返回值为非零值。  如果函数失败，则返回值为零。 注解 由于应用程序可以将矩形用于不同的用途，因此矩形函数不使用显式度量单位。 相反，所有矩形坐标和维度都以有符号的逻辑值提供。 映射模式和使用矩形的函数确定度量单位。 示例 有关示例，请参阅 [使用矩形](https://learn.microsoft.com/zh-cn/windows/desktop/gdi/using-rectangles)。 要求  |  |  | | --- | --- | | **标头** | winuser.h (包括 Windows.h) | | **Library** | User32.lib | | **DLL** | User32.dll |  另请参阅 [CopyRect](https://learn.microsoft.com/zh-cn/windows/desktop/api/winuser/nf-winuser-copyrect)  [矩形](https://learn.microsoft.com/zh-cn/windows/desktop/api/windef/ns-windef-rect)  [矩形函数](https://learn.microsoft.com/zh-cn/windows/desktop/gdi/rectangle-functions)  [矩形概述](https://learn.microsoft.com/zh-cn/windows/desktop/gdi/rectangles)  [SetRectEmpty](https://learn.microsoft.com/zh-cn/windows/desktop/api/winuser/nf-winuser-setrectempty) |

## CopyRect函数的用法

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **CopyRect** 函数将一个矩形的坐标复制到另一个矩形。 语法 C++  BOOL CopyRect(  [out] LPRECT lprcDst,  [in] const RECT \*lprcSrc  ); 参数 [out] lprcDst  指向接收源矩形逻辑坐标的 [RECT](https://learn.microsoft.com/zh-cn/windows/desktop/api/windef/ns-windef-rect) 结构的指针。  [in] lprcSrc  指向 [RECT](https://learn.microsoft.com/zh-cn/windows/desktop/api/windef/ns-windef-rect) 结构的指针，该结构将按逻辑单元复制其坐标。 返回值 如果该函数成功，则返回值为非零值。  如果函数失败，则返回值为零。 注解 由于应用程序可以将矩形用于不同的用途，因此矩形函数不使用显式度量单位。 相反，所有矩形坐标和维度都以有符号的逻辑值提供。 映射模式和使用矩形的函数确定度量单位。 示例 有关示例，请参阅 [使用矩形](https://learn.microsoft.com/zh-cn/windows/desktop/gdi/using-rectangles)。 要求  |  |  | | --- | --- | | **标头** | winuser.h (包括 Windows.h) | | **Library** | User32.lib | | **DLL** | User32.dll |  另请参阅 [矩形](https://learn.microsoft.com/zh-cn/windows/desktop/api/windef/ns-windef-rect)  [矩形函数](https://learn.microsoft.com/zh-cn/windows/desktop/gdi/rectangle-functions)  [矩形概述](https://learn.microsoft.com/zh-cn/windows/desktop/gdi/rectangles)  [SetRect](https://learn.microsoft.com/zh-cn/windows/desktop/api/winuser/nf-winuser-setrect)  [SetRectEmpty](https://learn.microsoft.com/zh-cn/windows/desktop/api/winuser/nf-winuser-setrectempty) |

## OffsetRect函数的用法

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **OffsetRect** 函数按指定的偏移量移动指定的矩形。 语法 C++复制  BOOL OffsetRect(  [in, out] LPRECT lprc,  [in] int dx,  [in] int dy  ); 参数 [in, out] lprc  指向 [RECT](https://learn.microsoft.com/zh-cn/windows/desktop/api/windef/ns-windef-rect) 结构的指针，该结构包含要移动的矩形的逻辑坐标。  [in] dx  指定向左或向右移动矩形的量。 此参数必须是负值才能将矩形向左移动。  [in] dy  指定向上或向下移动矩形的量。 此参数必须是负值才能向上移动矩形。 返回值 如果该函数成功，则返回值为非零值。  如果函数失败，则返回值为零。 注解 由于应用程序可以将矩形用于不同的用途，因此矩形函数不使用显式度量单位。 相反，所有矩形坐标和维度都以有符号的逻辑值提供。 映射模式和使用矩形的函数确定度量单位。 示例 有关示例，请参阅 [使用矩形](https://learn.microsoft.com/zh-cn/windows/desktop/gdi/using-rectangles)。 要求  |  |  | | --- | --- | | **标头** | winuser.h (包括 Windows.h) | | **Library** | User32.lib | | **DLL** | User32.dll |  另请参阅 [InflateRect](https://learn.microsoft.com/zh-cn/windows/desktop/api/winuser/nf-winuser-inflaterect)  [IntersectRect](https://learn.microsoft.com/zh-cn/windows/desktop/api/winuser/nf-winuser-intersectrect)  [矩形](https://learn.microsoft.com/zh-cn/windows/desktop/api/windef/ns-windef-rect)  [矩形函数](https://learn.microsoft.com/zh-cn/windows/desktop/gdi/rectangle-functions)  [矩形概述](https://learn.microsoft.com/zh-cn/windows/desktop/gdi/rectangles)  [UnionRect](https://learn.microsoft.com/zh-cn/windows/desktop/api/winuser/nf-winuser-unionrect) |

## InflateRect函数的用法

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **InflateRect** 函数增加或减小指定矩形的宽度和高度。 **InflateRect** 函数将 *-dx* 单位添加到矩形的左端，将 *dx* 添加到矩形的右端，*将 -dy* 单位添加到顶部，*将 dy* 添加到底部。 *dx* 和 *dy* 参数是有符号值;正值会增加宽度和高度，负值会减小宽度和高度。 语法 C++复制  BOOL InflateRect(  [in, out] LPRECT lprc,  [in] int dx,  [in] int dy  ); 参数 [in, out] lprc  指向大小增加或减小的 [RECT](https://learn.microsoft.com/zh-cn/windows/desktop/api/windef/ns-windef-rect) 结构的指针。  [in] dx  增加或减小矩形宽度的量。 此参数必须为负数才能减小宽度。  [in] dy  增加或减少矩形高度的量。 此参数必须为负值才能减小高度。 返回值 如果该函数成功，则返回值为非零值。  如果函数失败，则返回值为零。 注解 由于应用程序可以将矩形用于不同的目的，因此矩形函数不使用显式度量单位。 相反，所有矩形坐标和维度都以带符号的逻辑值提供。 使用矩形的映射模式和函数确定度量单位。 要求  |  |  | | --- | --- | | **标头** | winuser.h (包括 Windows.h) | | **Library** | User32.lib | | **DLL** | User32.dll |  另请参阅 [IntersectRect](https://learn.microsoft.com/zh-cn/windows/desktop/api/winuser/nf-winuser-intersectrect)  [OffsetRect](https://learn.microsoft.com/zh-cn/windows/desktop/api/winuser/nf-winuser-offsetrect)  [矩形](https://learn.microsoft.com/zh-cn/windows/desktop/api/windef/ns-windef-rect)  [矩形函数](https://learn.microsoft.com/zh-cn/windows/desktop/gdi/rectangle-functions)  [矩形概述](https://learn.microsoft.com/zh-cn/windows/desktop/gdi/rectangles)  [UnionRect](https://learn.microsoft.com/zh-cn/windows/desktop/api/winuser/nf-winuser-unionrect) |

## SetRectEmpty函数的用法

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **SetRectEmpty** 函数创建一个空矩形，其中所有坐标都设置为零。 语法 C++复制  BOOL SetRectEmpty(  [out] LPRECT lprc  ); 参数 [out] lprc  指向包含矩形坐标的 [RECT](https://learn.microsoft.com/zh-cn/windows/desktop/api/windef/ns-windef-rect) 结构的指针。 返回值 如果该函数成功，则返回值为非零值。  如果函数失败，则返回值为零。 注解 由于应用程序可以将矩形用于不同的用途，因此矩形函数不使用显式度量单位。 相反，所有矩形坐标和维度都以有符号的逻辑值提供。 映射模式和使用矩形的函数确定度量单位。 示例 有关示例，请参阅 [使用矩形](https://learn.microsoft.com/zh-cn/windows/desktop/gdi/using-rectangles)。 要求  |  |  | | --- | --- | | **标头** | winuser.h (包括 Windows.h) | | **Library** | User32.lib | | **DLL** | User32.dll |  另请参阅 [CopyRect](https://learn.microsoft.com/zh-cn/windows/desktop/api/winuser/nf-winuser-copyrect)  [矩形](https://learn.microsoft.com/zh-cn/windows/desktop/api/windef/ns-windef-rect)  [矩形函数](https://learn.microsoft.com/zh-cn/windows/desktop/gdi/rectangle-functions)  [矩形概述](https://learn.microsoft.com/zh-cn/windows/desktop/gdi/rectangles)  [SetRect](https://learn.microsoft.com/zh-cn/windows/desktop/api/winuser/nf-winuser-setrect) |

## IsRectEmpty函数的用法

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **IsRectEmpty** 函数确定指定的矩形是否为空。 空矩形是没有区域的矩形;也就是说，右侧的坐标小于或等于左侧的坐标，或者底部的坐标小于或等于顶部的坐标。 语法 C++  BOOL IsRectEmpty(  [in] const RECT \*lprc  ); 参数 [in] lprc  指向包含矩形逻辑坐标的 [RECT](https://learn.microsoft.com/zh-cn/windows/desktop/api/windef/ns-windef-rect) 结构的指针。 返回值 如果矩形为空，则返回值为非零值。  如果矩形不为空，则返回值为零。 注解 由于应用程序可以将矩形用于不同的目的，因此矩形函数不使用显式度量单位。 相反，所有矩形坐标和维度都以带符号的逻辑值提供。 使用矩形的映射模式和函数确定度量单位。 示例 有关示例，请参阅 [使用矩形](https://learn.microsoft.com/zh-cn/windows/desktop/gdi/using-rectangles)。 要求  |  |  | | --- | --- | | **标头** | winuser.h (包括 Windows.h) | | **Library** | User32.lib | | **DLL** | User32.dll |  另请参阅 [EqualRect](https://learn.microsoft.com/zh-cn/windows/desktop/api/winuser/nf-winuser-equalrect)  [PtInRect](https://learn.microsoft.com/zh-cn/windows/desktop/api/winuser/nf-winuser-ptinrect)  [矩形](https://learn.microsoft.com/zh-cn/windows/desktop/api/windef/ns-windef-rect)  [矩形函数](https://learn.microsoft.com/zh-cn/windows/desktop/gdi/rectangle-functions)  [矩形概述](https://learn.microsoft.com/zh-cn/windows/desktop/gdi/rectangles) |

## IntersectRect函数的用法

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **IntersectRect** 函数计算两个源矩形的交集，并将交集矩形的坐标放入目标矩形。 如果源矩形不相交，则 (一个空矩形，其中所有坐标都设置为零，) 放置在目标矩形中。 语法 C++复制  BOOL IntersectRect(  [out] LPRECT lprcDst,  [in] const RECT \*lprcSrc1,  [in] const RECT \*lprcSrc2  ); 参数 [out] lprcDst  指向 [RECT](https://learn.microsoft.com/zh-cn/windows/desktop/api/windef/ns-windef-rect) 结构的指针，用于接收 *lprcSrc1* 和 *lprcSrc2* 参数指向的矩形的交集。 此参数不能为 **NULL**。  [in] lprcSrc1  指向包含第一个源矩形的 [RECT](https://learn.microsoft.com/zh-cn/windows/desktop/api/windef/ns-windef-rect) 结构的指针。  [in] lprcSrc2  指向包含第二个源矩形的 [RECT](https://learn.microsoft.com/zh-cn/windows/desktop/api/windef/ns-windef-rect) 结构的指针。 返回值 如果矩形相交，则返回值为非零值。  如果矩形不相交，则返回值为零。 注解 由于应用程序可以将矩形用于不同的目的，因此矩形函数不使用显式度量单位。 相反，所有矩形坐标和维度都以带符号的逻辑值提供。 使用矩形的映射模式和函数确定度量单位。 示例 有关示例，请参阅 [使用矩形](https://learn.microsoft.com/zh-cn/windows/desktop/gdi/using-rectangles)。 要求  |  |  | | --- | --- | | **标头** | winuser.h (包括 Windows.h) | | **Library** | User32.lib | | **DLL** | User32.dll |  另请参阅 [InflateRect](https://learn.microsoft.com/zh-cn/windows/desktop/api/winuser/nf-winuser-inflaterect)  [OffsetRect](https://learn.microsoft.com/zh-cn/windows/desktop/api/winuser/nf-winuser-offsetrect)  [矩形](https://learn.microsoft.com/zh-cn/windows/desktop/api/windef/ns-windef-rect)  [矩形函数](https://learn.microsoft.com/zh-cn/windows/desktop/gdi/rectangle-functions)  [矩形概述](https://learn.microsoft.com/zh-cn/windows/desktop/gdi/rectangles)  [UnionRect](https://learn.microsoft.com/zh-cn/windows/desktop/api/winuser/nf-winuser-unionrect) |

## UnionRect函数的用法

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **UnionRect** 函数创建两个矩形的联合。 并集是包含两个源矩形的最小矩形。 语法 C++复制  BOOL UnionRect(  [out] LPRECT lprcDst,  [in] const RECT \*lprcSrc1,  [in] const RECT \*lprcSrc2  ); 参数 [out] lprcDst  指向 [RECT](https://learn.microsoft.com/zh-cn/windows/desktop/api/windef/ns-windef-rect) 结构的指针，该结构将接收包含 *lprcSrc1 和 lprcSrc2* 参数指向的矩形 *的* 矩形。  [in] lprcSrc1  指向包含第一个源矩形的 [RECT](https://learn.microsoft.com/zh-cn/windows/desktop/api/windef/ns-windef-rect) 结构的指针。  [in] lprcSrc2  指向包含第二个源矩形的 [RECT](https://learn.microsoft.com/zh-cn/windows/desktop/api/windef/ns-windef-rect) 结构的指针。 返回值 如果指定的 结构包含非空矩形，则返回值为非零值。  如果指定的结构不包含非空矩形，则返回值为零。 注解 系统忽略空矩形的尺寸，即所有坐标都设置为零的矩形，使其没有高度或没有宽度。  由于应用程序可以将矩形用于不同的用途，因此矩形函数不使用显式度量单位。 相反，所有矩形坐标和维度都以有符号的逻辑值提供。 映射模式和使用矩形的函数确定度量单位。 要求  |  |  | | --- | --- | | **标头** | winuser.h (包括 Windows.h) | | **Library** | User32.lib | | **DLL** | User32.dll |  另请参阅 [InflateRect](https://learn.microsoft.com/zh-cn/windows/desktop/api/winuser/nf-winuser-inflaterect)  [IntersectRect](https://learn.microsoft.com/zh-cn/windows/desktop/api/winuser/nf-winuser-intersectrect)  [OffsetRect](https://learn.microsoft.com/zh-cn/windows/desktop/api/winuser/nf-winuser-offsetrect)  [矩形](https://learn.microsoft.com/zh-cn/windows/desktop/api/windef/ns-windef-rect)  [矩形函数](https://learn.microsoft.com/zh-cn/windows/desktop/gdi/rectangle-functions)  [矩形概述](https://learn.microsoft.com/zh-cn/windows/desktop/gdi/rectangles) |

## PtInRect函数的用法

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **PtInRect** 函数确定指定的点是否位于指定的矩形内。 如果点位于左侧或顶部，或者位于所有四个边内，则点位于矩形内。 右侧或底部的点被视为矩形外部的点。 语法 C++复制  BOOL PtInRect(  [in] const RECT \*lprc,  [in] POINT pt  ); 参数 [in] lprc  指向包含指定矩形的 [RECT](https://learn.microsoft.com/zh-cn/windows/desktop/api/windef/ns-windef-rect) 结构的指针。  [in] pt  包含指定点的 [POINT](https://learn.microsoft.com/zh-cn/windows/win32/api/windef/ns-windef-point) 结构。 返回值 如果指定的点位于矩形内，则返回值为非零值。  如果指定的点不在矩形内，则返回值为零。 注解 在调用 **PtInRect** 之前，必须规范化矩形。 也就是说，lprc.right 必须大于 lprc.left，lprc.bottom 必须大于 lprc.top。 如果未规范化矩形，则永远不会将点视为矩形内部。  由于应用程序可以将矩形用于不同的用途，因此矩形函数不使用显式度量单位。 相反，所有矩形坐标和维度都以有符号的逻辑值提供。 映射模式和使用矩形的函数确定度量单位。 示例 有关示例，请参阅 [使用矩形](https://learn.microsoft.com/zh-cn/windows/desktop/gdi/using-rectangles)。 要求  |  |  | | --- | --- | | **标头** | winuser.h (包括 Windows.h) | | **Library** | User32.lib | | **DLL** | User32.dll |  另请参阅 [EqualRect](https://learn.microsoft.com/zh-cn/windows/desktop/api/winuser/nf-winuser-equalrect)  [IsRectEmpty](https://learn.microsoft.com/zh-cn/windows/desktop/api/winuser/nf-winuser-isrectempty)  [点](https://learn.microsoft.com/zh-cn/windows/win32/api/windef/ns-windef-point)  [矩形](https://learn.microsoft.com/zh-cn/windows/desktop/api/windef/ns-windef-rect)  [矩形函数](https://learn.microsoft.com/zh-cn/windows/desktop/gdi/rectangle-functions)  [矩形概述](https://learn.microsoft.com/zh-cn/windows/desktop/gdi/rectangles) |

## RECT结构体的用法

|  |  |  |
| --- | --- | --- |
| RECT 结构通过矩形的左上角和右下角的坐标来定义矩形。 语法 C++复制  typedef struct tagRECT {  LONG left;  LONG top;  LONG right;  LONG bottom;  } RECT, \*PRECT, \*NPRECT, \*LPRECT; 成员 left  指定矩形左上角的 *x* 坐标。  top  指定矩形左上角的 *y* 坐标。  right  指定矩形右下角的 *x* 坐标。  bottom  指定矩形右下角的 *y* 坐标。 注解 RECT 结构与 [RECTL](https://learn.microsoft.com/zh-cn/windows/desktop/api/windef/ns-windef-rectl) 结构相同。 要求  |  |  | | --- | --- | | **Header** | windef.h (包括 Windows.h) |  另请参阅 [RECTL](https://learn.microsoft.com/zh-cn/windows/desktop/api/windef/ns-windef-rectl) |

# 演练

## 1.新建一个win32项目，取名Lesson46-rect-demo

|  |
| --- |
|  |

## 2.找到WndProc函数，先定义一个RECT结构体变量，然后定位到WM\_PAINT消息处理代码，绘制一个填充矩形。

|  |
| --- |
|  |
|  |

### 效果

|  |
| --- |
|  |

## 3.我们来学习PtInRect函数，这个函数可以判断一个点是否在矩形里面，这里需要响应WM\_LBUTTONDOWN学习来获取点，然后判断它是否在矩形里面，需要定义一个点结构的变量来保存鼠标点击的点

|  |
| --- |
|  |
|  |

### 效果

|  |
| --- |
|  |

## 4.下面我们来学习FrameRect函数,就是用自己定义的画刷来绘制矩形的边框

|  |
| --- |
|  |

### 效果

|  |
| --- |
|  |

### 注意：和Rectangle函数不一样，这个函数用画刷来绘制矩形而不是画笔，而且画刷不用选入设备描述

## 5.我们可以把一个矩形反转，把矩形的填充颜色取反描边颜色也取反，为了方便对比，我们把矩形设置为红色边框和蓝色填充

|  |
| --- |
|  |
|  |

### 然后我们调用InvertRect函数，就会得到一个蓝色边框和黄色填充的矩形

|  |
| --- |
|  |

### 效果如下

|  |
| --- |
|  |

## 6.我们可以使用InflateRect把矩形扩充，我们可以在填充矩形之前把他扩充

|  |
| --- |
|  |

### 效果

|  |
| --- |
|  |

### 注意：如果cx和cy为负数，矩形就会变小

## 7.我们使用OffsetRect函数来移动矩形的位置，我们把他往右边移动

|  |
| --- |
|  |

### 效果

|  |
| --- |
|  |

## 8.除了InflateRect外，我们也可以使用SetRect函数来改变矩形的大小

|  |
| --- |
|  |

### 效果

|  |
| --- |
|  |

## 9.可以把一个矩形的坐标复制给另外一个矩形，使用CopyRect函数，这里需要定义一个新矩形变量，然后我们可以复制矩形到这个变量，然后把平移250像素再提出紫色

|  |
| --- |
|  |

### 效果

|  |
| --- |
|  |

## 10.下面我们来学习求两个矩形的交集，我们先调整一下第二个矩形的偏移量，使得它和第一个矩形有相交

|  |
| --- |
|  |
|  |

## 11.然后我们来求这两个矩形的交集，需要一个矩形来保存数据

|  |
| --- |
|  |
|  |

### 效果

|  |
| --- |
|  |

## 12.可以使用UnionRect函数把2个矩形做一个并集，我们先定义一个矩形来存储并集

|  |
| --- |
|  |

### 然后我们来做并集

|  |
| --- |
|  |

### 效果

|  |
| --- |
|  |

## 13如果你想把一个矩形的全部坐标设置为0，你除了可以一个一个来设置以外，还可以调用SetRectEmpty函数一次到位完成这个功能，如果我们在填充上面的并集矩形之前把他清空，那么后面的填充语句就需要然后作用

|  |
| --- |
|  |

### 效果，因为并集矩形被清空，填充没有效果，就可以看到我们前面填充的效果

|  |
| --- |
|  |

## 14.可以调用IsRectEmpty函数来判断矩形是否为空，所谓的空矩形就是指这个矩形的四个坐标值都为0的情况

|  |
| --- |
|  |

### 效果

|  |
| --- |
|  |

# 这一节的学习到此为止，完整代码如下

## Lesson46-rect-demo.cpp

|  |
| --- |
| // Lesson46-rect-demo.cpp : 定义应用程序的入口点。  //  #include "stdafx.h"  #include "Lesson46-rect-demo.h"  #define MAX\_LOADSTRING 100  // 全局变量:  HINSTANCE hInst; // 当前实例  TCHAR szTitle[MAX\_LOADSTRING]; // 标题栏文本  TCHAR szWindowClass[MAX\_LOADSTRING]; // 主窗口类名  // 此代码模块中包含的函数的前向声明:  ATOM MyRegisterClass(HINSTANCE hInstance);  BOOL InitInstance(HINSTANCE, int);  LRESULT CALLBACK WndProc(HWND, UINT, WPARAM, LPARAM);  INT\_PTR CALLBACK About(HWND, UINT, WPARAM, LPARAM);  int APIENTRY \_tWinMain(HINSTANCE hInstance,  HINSTANCE hPrevInstance,  LPTSTR lpCmdLine,  int nCmdShow)  {  UNREFERENCED\_PARAMETER(hPrevInstance);  UNREFERENCED\_PARAMETER(lpCmdLine);  // TODO: 在此放置代码。  MSG msg;  HACCEL hAccelTable;  // 初始化全局字符串  LoadString(hInstance, IDS\_APP\_TITLE, szTitle, MAX\_LOADSTRING);  LoadString(hInstance, IDC\_LESSON46RECTDEMO, szWindowClass, MAX\_LOADSTRING);  MyRegisterClass(hInstance);  // 执行应用程序初始化:  if (!InitInstance (hInstance, nCmdShow))  {  return FALSE;  }  hAccelTable = LoadAccelerators(hInstance, MAKEINTRESOURCE(IDC\_LESSON46RECTDEMO));  // 主消息循环:  while (GetMessage(&msg, NULL, 0, 0))  {  if (!TranslateAccelerator(msg.hwnd, hAccelTable, &msg))  {  TranslateMessage(&msg);  DispatchMessage(&msg);  }  }  return (int) msg.wParam;  }  //  // 函数: MyRegisterClass()  //  // 目的: 注册窗口类。  //  // 注释:  //  // 仅当希望  // 此代码与添加到 Windows 95 中的“RegisterClassEx”  // 函数之前的 Win32 系统兼容时，才需要此函数及其用法。调用此函数十分重要，  // 这样应用程序就可以获得关联的  // “格式正确的”小图标。  //  ATOM MyRegisterClass(HINSTANCE hInstance)  {  WNDCLASSEX wcex;  wcex.cbSize = sizeof(WNDCLASSEX);  wcex.style = CS\_HREDRAW | CS\_VREDRAW;  wcex.lpfnWndProc = WndProc;  wcex.cbClsExtra = 0;  wcex.cbWndExtra = 0;  wcex.hInstance = hInstance;  wcex.hIcon = LoadIcon(hInstance, MAKEINTRESOURCE(IDI\_LESSON46RECTDEMO));  wcex.hCursor = LoadCursor(NULL, IDC\_ARROW);  wcex.hbrBackground = (HBRUSH)(COLOR\_WINDOW+1);  wcex.lpszMenuName = MAKEINTRESOURCE(IDC\_LESSON46RECTDEMO);  wcex.lpszClassName = szWindowClass;  wcex.hIconSm = LoadIcon(wcex.hInstance, MAKEINTRESOURCE(IDI\_SMALL));  return RegisterClassEx(&wcex);  }  //  // 函数: InitInstance(HINSTANCE, int)  //  // 目的: 保存实例句柄并创建主窗口  //  // 注释:  //  // 在此函数中，我们在全局变量中保存实例句柄并  // 创建和显示主程序窗口。  //  BOOL InitInstance(HINSTANCE hInstance, int nCmdShow)  {  HWND hWnd;  hInst = hInstance; // 将实例句柄存储在全局变量中  hWnd = CreateWindow(szWindowClass, szTitle, WS\_OVERLAPPEDWINDOW,  CW\_USEDEFAULT, 0, CW\_USEDEFAULT, 0, NULL, NULL, hInstance, NULL);  if (!hWnd)  {  return FALSE;  }  ShowWindow(hWnd, nCmdShow);  UpdateWindow(hWnd);  return TRUE;  }  //  // 函数: WndProc(HWND, UINT, WPARAM, LPARAM)  //  // 目的: 处理主窗口的消息。  //  // WM\_COMMAND - 处理应用程序菜单  // WM\_PAINT - 绘制主窗口  // WM\_DESTROY - 发送退出消息并返回  //  //  LRESULT CALLBACK WndProc(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam)  {  int wmId, wmEvent;  PAINTSTRUCT ps;  HDC hdc;  RECT rect = {100,100,300,200};  HBRUSH brush,OrgBrush;//填充矩形的画刷  POINT pt;//保存鼠标点击的点的坐标  RECT rect2;  RECT rectIntersect; //交集矩形  RECT rectUnion; //并集矩形  switch (message)  {  case WM\_LBUTTONDOWN:  pt.x = LOWORD(lParam);  pt.y =HIWORD(lParam);  if(PtInRect(&rect,pt))  {  MessageBox(hWnd,\_T("point inside Rect"),\_T("Result"),0);  }  break;  case WM\_COMMAND:  wmId = LOWORD(wParam);  wmEvent = HIWORD(wParam);  // 分析菜单选择:  switch (wmId)  {  case IDM\_ABOUT:  DialogBox(hInst, MAKEINTRESOURCE(IDD\_ABOUTBOX), hWnd, About);  break;  case IDM\_EXIT:  DestroyWindow(hWnd);  break;  default:  return DefWindowProc(hWnd, message, wParam, lParam);  }  break;  case WM\_PAINT:  hdc = BeginPaint(hWnd, &ps);  // TODO: 在此添加任意绘图代码...    brush = CreateSolidBrush(RGB(255,0,0));//红色边框  //把矩形扩充，x方向增加50，y方向也增加50  //InflateRect(&rect,50,50);  //偏移矩形的位置  //OffsetRect(&rect,300,0);  //设置矩形的大小，需要设置左上角和右下角的坐标  //SetRect(&rect,10,10,400,300);  //绘制填充矩形  FillRect(hdc,&rect,CreateSolidBrush(RGB(0,0,255)));//用蓝色画刷填充  //绘制边框  //FrameRect(hdc,&rect,brush);  //把矩形填充颜色取反,描边颜色也取反  //InvertRect(hdc,&rect);  //复制矩形数据  CopyRect(&rect2,&rect);  OffsetRect(&rect2,50,50);  FillRect(hdc,&rect2,CreateSolidBrush(RGB(255,0,255))); //填充紫色  //求这两个矩形的相交部分  IntersectRect(&rectIntersect,&rect,&rect2);  //给相交部分填充青色  FillRect(hdc,&rectIntersect,CreateSolidBrush(RGB(0,255,255)));  //合并矩形,虽然两个矩形不是对齐的，合并后会变为一个规整的矩形  UnionRect(&rectUnion,&rect,&rect2);  //在填充并集矩形之前清空矩形  SetRectEmpty(&rectUnion);  //矩形被清空后填充就没有效果  FillRect(hdc,&rectUnion,CreateSolidBrush(RGB(255,255,155)));  //判断矩形是否为空  if(IsRectEmpty(&rectUnion))  {  MessageBox(hWnd,\_T("合并矩形被清空"),\_T("提示"),0);  }  EndPaint(hWnd, &ps);    break;  case WM\_DESTROY:  PostQuitMessage(0);  break;  default:  return DefWindowProc(hWnd, message, wParam, lParam);  }  return 0;  }  // “关于”框的消息处理程序。  INT\_PTR CALLBACK About(HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam)  {  UNREFERENCED\_PARAMETER(lParam);  switch (message)  {  case WM\_INITDIALOG:  return (INT\_PTR)TRUE;  case WM\_COMMAND:  if (LOWORD(wParam) == IDOK || LOWORD(wParam) == IDCANCEL)  {  EndDialog(hDlg, LOWORD(wParam));  return (INT\_PTR)TRUE;  }  break;  }  return (INT\_PTR)FALSE;  } |