# 学习大纲

|  |
| --- |
|  |

## LoadBitmap函数的用法

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [**LoadBitmap** 可用于“要求”部分中指定的操作系统。 它可能在后续版本中变更或不可用。 请改用 [LoadImage](https://learn.microsoft.com/zh-cn/windows/desktop/api/winuser/nf-winuser-loadimagea) 和 [DrawFrameControl](https://learn.microsoft.com/zh-cn/windows/desktop/api/winuser/nf-winuser-drawframecontrol)。]  **LoadBitmap** 函数从模块的可执行文件加载指定的位图资源。 语法 C++复制  HBITMAP LoadBitmapA(  [in] HINSTANCE hInstance,  [in] LPCSTR lpBitmapName  ); 参数 [in] hInstance  模块实例的句柄，该模块的可执行文件包含要加载的位图。  [in] lpBitmapName  指向以 null 结尾的字符串的指针，该字符串包含要加载的位图资源的名称。 或者，此参数可以包含低序字中的资源标识符和高序字中的零。 [MAKEINTRESOURCE](https://learn.microsoft.com/zh-cn/windows/desktop/api/winuser/nf-winuser-makeintresourcea) 宏可用于创建此值。 返回值 如果函数成功，则返回值是指定位图的句柄。  如果函数失败，则返回值为 NULL。 注解 如果 *lpBitmapName* 参数指向的位图不存在或内存不足，无法加载位图，则函数将失败。  **LoadBitmap** 创建显示器的兼容位图，该位图不能在打印机上选择。 若要将可以选择的位图加载到打印机，请调用 [LoadImage](https://learn.microsoft.com/zh-cn/windows/desktop/api/winuser/nf-winuser-loadimagea) 并指定LR\_CREATEDIBSECTION以创建 DIB 节。 可以为任何设备选择 DIB 部分。  应用程序可以使用 **LoadBitmap** 函数访问预定义的位图。 为此，应用程序必须将 *hInstance* 参数设置为 **NULL** ，并将 *lpBitmapName* 参数设置为以下值之一。  展开表   |  |  | | --- | --- | | **位图名称** | **位图名称** | | OBM\_BTNCORNERS | OBM\_OLD\_RESTORE | | OBM\_BTSIZE | OBM\_OLD\_RGARROW | | OBM\_CHECK | OBM\_OLD\_UPARROW | | OBM\_CHECKBOXES | OBM\_OLD\_ZOOM | | OBM\_CLOSE | OBM\_REDUCE | | OBM\_COMBO | OBM\_REDUCED | | OBM\_DNARROW | OBM\_RESTORE | | OBM\_DNARROWD | OBM\_RESTORED | | OBM\_DNARROWI | OBM\_RGARROW | | OBM\_LFARROW | OBM\_RGARROWD | | OBM\_LFARROWD | OBM\_RGARROWI | | OBM\_LFARROWI | OBM\_SIZE | | OBM\_MNARROW | OBM\_UPARROW | | OBM\_OLD\_CLOSE | OBM\_UPARROWD | | OBM\_OLD\_DNARROW | OBM\_UPARROWI | | OBM\_OLD\_LFARROW | OBM\_ZOOM | | OBM\_OLD\_REDUCE | OBM\_ZOOMD |     以 OBM\_OLD 开头的位图名称表示早于 3.0 的 16 位版本的 Windows 使用的位图。  若要使应用程序使用任何OBM\_常量，必须在包含 Windows.h 头文件之前定义常量 OEMRESOURCE。  应用程序必须调用 [DeleteObject](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-deleteobject) 函数才能删除 **LoadBitmap** 函数返回的每个位图句柄。 示例 有关示例，请参阅 [Using Menus](https://learn.microsoft.com/zh-cn/windows/desktop/menurc/using-menus) 中的 Menu-Item 位图示例。  **备注**  winuser.h 标头将 LoadBitmap 定义为别名，该别名根据 UNICODE 预处理器常量的定义自动选择此函数的 ANSI 或 Unicode 版本。 将非特定编码别名与非非特定编码的代码混合使用可能会导致不匹配，从而导致编译或运行时错误。 有关详细信息，请参阅 [**函数原型的约定**](https://learn.microsoft.com/zh-cn/windows/win32/intl/conventions-for-function-prototypes)。 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-draw-l1-1-1 ( |  另请参阅 [位图函数](https://learn.microsoft.com/zh-cn/windows/desktop/gdi/bitmap-functions)  [位图概述](https://learn.microsoft.com/zh-cn/windows/desktop/gdi/bitmaps)  [CreateBitmap](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-createbitmap)  [DeleteObject](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-deleteobject)  [DrawFrameControl](https://learn.microsoft.com/zh-cn/windows/desktop/api/winuser/nf-winuser-drawframecontrol)  [LoadCursor](https://learn.microsoft.com/zh-cn/windows/desktop/api/winuser/nf-winuser-loadcursora)  [LoadIcon](https://learn.microsoft.com/zh-cn/windows/desktop/api/winuser/nf-winuser-loadicona)  [LoadImage](https://learn.microsoft.com/zh-cn/windows/desktop/api/winuser/nf-winuser-loadimagea)  [MAKEINTRESOURCE](https://learn.microsoft.com/zh-cn/windows/desktop/api/winuser/nf-winuser-makeintresourcea) |

## CreatePatternBrush函数的用法

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **CreatePatternBrush** 函数使用指定的位图模式创建逻辑画笔。 位图可以是由 **CreateDIBSection** 函数创建的 DIB 节位图，也可以是依赖于设备的位图。 语法 C++  HBRUSH CreatePatternBrush(  [in] HBITMAP hbm  ); 参数 [in] hbm  要用于创建逻辑画笔的位图的句柄。 返回值 如果函数成功，则返回值标识逻辑画笔。  如果函数失败，则返回值为 NULL。 注解 图案画笔是系统用于绘制填充形状内部的位图。  应用程序通过调用 **CreatePatternBrush** 创建画笔后，可以通过调用 [SelectObject](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-selectobject) 函数在任何设备上下文中选择该画笔。  可以使用 [DeleteObject](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-deleteobject) 函数删除图案画笔，而不会影响关联的位图。 因此，可以使用此位图创建任意数量的图案画笔。  使用单色 (每像素 1 位) 位图创建的画笔具有绘制到的设备上下文的文本和背景色。 用当前文本颜色绘制由 0 位表示的像素;用当前背景色绘制由 1 位表示的像素。  **Icm：** 创建画笔时不执行任何颜色。 但是，当画笔被选入启用了 ICM 的设备上下文时，将执行颜色管理。 示例 有关示例，请参阅 [使用画笔](https://learn.microsoft.com/zh-cn/windows/desktop/gdi/using-brushes)。 要求  |  |  | | --- | --- | | **标头** | wingdi.h (包括 Windows.h) | | **Library** | Gdi32.lib | | **DLL** | Gdi32.dll |  另请参阅 [画笔函数](https://learn.microsoft.com/zh-cn/windows/desktop/gdi/brush-functions)  [画笔概述](https://learn.microsoft.com/zh-cn/windows/desktop/gdi/brushes)  [CreateBitmap](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-createbitmap)  [CreateBitmapIndirect](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-createbitmapindirect)  [CreateCompatibleBitmap](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-createcompatiblebitmap)  [CreateDIBPatternBrush](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-createdibpatternbrush)  [CreateDIBPatternBrushPt](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-createdibpatternbrushpt)  [CreateDIBSection](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-createdibsection)  [CreateHatchBrush](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-createhatchbrush)  [DeleteObject](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-deleteobject)  [GetBrushOrgEx](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-getbrushorgex)  [LoadBitmap](https://learn.microsoft.com/zh-cn/windows/desktop/api/winuser/nf-winuser-loadbitmapa)  [SelectObject](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-selectobject)  [SetBrushOrgEx](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-setbrushorgex) |

### 可以调用这个函数来创建一个位图画刷

## GetObject函数的用法

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **GetObject** 函数检索指定图形对象的信息。 语法 C++  int GetObject(  [in] HANDLE h,  [in] int c,  [out] LPVOID pv  ); 参数 [in] h  感兴趣的图形对象的句柄。 这可以是以下项之一的句柄：逻辑位图、画笔、字体、调色板、笔或通过调用 [CreateDIBSection](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-createdibsection) 函数创建的独立于设备的位图。  [in] c  要写入缓冲区的信息字节数。  [out] pv  指向接收指定图形对象相关信息的缓冲区的指针。  下表显示了缓冲区为可以使用 *hgdiobj* 指定的每种类型的图形对象接收的信息类型。  展开表   |  |  | | --- | --- | | **对象类型** | **写入缓冲区的数据** | | **HBITMAP** | [BITMAP](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/ns-wingdi-bitmap) | | **从**[**对 CreateDIBSection**](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-createdibsection)**的调用返回的 HBITMAP** | [DIBSECTION](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/ns-wingdi-dibsection)（如果 *cbBuffer* 设置为 sizeof (DIBSECTION) ）或 [BITMAP](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/ns-wingdi-bitmap)（如果 *cbBuffer* 设置为 sizeof (BITMAP)）。 | | **HPALETTE** | 逻辑调色板中条目数的 **WORD** 计数 | | **从对**[**ExtCreatePen**](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-extcreatepen)**的调用返回的 HPEN** | [EXTLOGPEN](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/ns-wingdi-extlogpen) | | **HPEN** | [LOGPEN](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/ns-wingdi-logpen) | | **HBRUSH** | [LOGBRUSH](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/ns-wingdi-logbrush) | | **HFONT** | [LOGFONT](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/ns-wingdi-logfonta) |     如果 *lpvObject* 参数为 **NULL**，则函数返回值是存储它写入指定图形对象的缓冲区的信息所需的字节数。  *lpvObject* 的地址必须位于 4 字节边界上;否则，**GetObject** 将失败。 返回值 如果函数成功，并且 *lpvObject* 是有效的指针，则返回值是存储在缓冲区中的字节数。  如果函数成功，并且 *lpvObject* 为 **NULL**，则返回值是保存函数将存储到缓冲区中的信息所需的字节数。  如果函数失败，则返回值为零。 注解 *lpvObject* 参数指向的缓冲区必须足够大，才能接收有关图形对象的信息。 根据图形对象，该函数使用 [BITMAP](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/ns-wingdi-bitmap)、 [DIBSECTION](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/ns-wingdi-dibsection)、 [EXTLOGPEN](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/ns-wingdi-extlogpen)、 [LOGBRUSH](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/ns-wingdi-logbrush)、 [LOGFONT](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/ns-wingdi-logfonta) 或 [LOGPEN](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/ns-wingdi-logpen) 结构，或者逻辑调色板) (表条目计数。  如果 *hgdiobj* 是通过调用 [CreateDIBSection](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-createdibsection) 创建的位图的句柄，并且指定的缓冲区足够大， **则 GetObject** 函数将返回 [DIBSECTION](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/ns-wingdi-dibsection) 结构。 此外，**DIBSECTION** 中包含的 [BITMAP](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/ns-wingdi-bitmap) 结构的 **bmBits** 成员将包含指向位图位值的指针。  如果 *hgdiobj* 是通过任何其他方式创建的位图的句柄， **则 GetObject** 仅返回位图的宽度、高度和颜色格式信息。 可以通过调用 [GetDIBits 或 GetBitmapBits](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-getdibits) 函数来获取位 [图的位](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-getbitmapbits) 值。  如果 *hgdiobj* 是逻辑调色板的句柄， **则 GetObject** 将检索一个 2 字节整数，该整数指定调色板中的条目数。 函数不会检索定义调色板的 [LOGPALETTE](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/ns-wingdi-logpalette) 结构。 若要检索有关调色板条目的信息，应用程序可以调用 [GetPaletteEntries](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-getpaletteentries) 函数。  如果 *hgdiobj* 是字体的句柄，则返回的 [LOGFONT](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/ns-wingdi-logfonta) 是用于创建字体的 **LOGFONT** 。 如果 Windows 由于无法表示精确的 **LOGFONT** 而不得不对字体进行一些内插，则内插不会反映在 **LOGFONT** 中。 例如，如果你要求提供不支持垂直绘制的字体的垂直版本， **则 LOGFONT** 指示该字体是垂直的，但 Windows 将水平绘制该字体。 示例 有关示例，请参阅 [存储映像](https://learn.microsoft.com/zh-cn/windows/desktop/gdi/storing-an-image)。 要求  |  |  | | --- | --- | | **标头** | wingdi.h (包括 Windows.h) | | **Library** | Gdi32.lib | | **DLL** | Gdi32.dll |  另请参阅 [BITMAP](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/ns-wingdi-bitmap)  [CreateDIBSection](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-createdibsection)  [DIBSECTION](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/ns-wingdi-dibsection)  [设备上下文函数](https://learn.microsoft.com/zh-cn/windows/desktop/gdi/device-context-functions)  [设备上下文概述](https://learn.microsoft.com/zh-cn/windows/desktop/gdi/device-contexts)  [EXTLOGPEN](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/ns-wingdi-extlogpen)  [GetBitmapBits](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-getbitmapbits)  [GetDIBits](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-getdibits)  [GetPaletteEntries](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-getpaletteentries)  [GetRegionData](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-getregiondata)  [LOGBRUSH](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/ns-wingdi-logbrush)  [LOGFONT](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/ns-wingdi-logfonta)  [LOGPALETTE](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/ns-wingdi-logpalette)  [LOGPEN](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/ns-wingdi-logpen) |

## CreateCompatibleDC函数的用法

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **CreateCompatibleDC** 函数创建与指定设备兼容的内存设备上下文 (DC) 。 语法 C++复制  HDC CreateCompatibleDC(  [in] HDC hdc  ); 参数 [in] hdc  现有 DC 的句柄。 如果此句柄为 **NULL**，则该函数将创建与应用程序的当前屏幕兼容的内存 DC。 返回值 如果函数成功，则返回值是内存 DC 的句柄。  如果函数失败，则返回值为 NULL。 注解 内存 DC 仅存在于内存中。 创建内存 DC 时，其显示表面正好是一个单色像素宽和一个单色像素高。 在应用程序可以使用内存 DC 进行绘制操作之前，它必须在 DC 中选择一个宽度和高度正确的位图。 若要选择 DC 中的位图，请使用 [CreateCompatibleBitmap](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-createcompatiblebitmap) 函数，指定所需的高度、宽度和颜色组织。  创建内存 DC 时，所有属性都设置为普通默认值。 内存 DC 可用作普通 DC。 可以设置属性;获取其属性的当前设置;并选择笔、画笔和区域。  **CreateCompatibleDC** 函数只能用于支持光栅操作的设备。 应用程序可以通过调用 [GetDeviceCaps](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-getdevicecaps) 函数来确定设备是否支持这些操作。  如果不再需要内存 DC，请调用 [DeleteDC](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-deletedc) 函数。 建议调用 **DeleteDC** 来删除 DC。 但是，也可以使用 HDC 调用 [DeleteObject](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-deleteobject) 来删除 DC。  如果 *hdc* 为 **NULL**，则调用 **CreateCompatibleDC** 的线程将拥有创建的 HDC。 当此线程被销毁时，HDC 不再有效。 因此，如果创建 HDC 并将其传递给另一个线程，然后退出第一个线程，则第二个线程将无法使用 HDC。  **Icm：** 如果为图像颜色管理 (ICM) 启用了传递给此函数的 DC，则由该函数创建的 DC 已启用 ICM。 源和目标颜色空间在 DC 中指定。 示例 有关示例，请参阅 [捕获图像](https://learn.microsoft.com/zh-cn/windows/desktop/gdi/capturing-an-image)。 要求  |  |  | | --- | --- | | **标头** | wingdi.h (包括 Windows.h) | | **Library** | Gdi32.lib | | **DLL** | Gdi32.dll |  另请参阅 [CreateCompatibleBitmap](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-createcompatiblebitmap)  [DeleteDC](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-deletedc)  [设备上下文函数](https://learn.microsoft.com/zh-cn/windows/desktop/gdi/device-context-functions)  [设备上下文概述](https://learn.microsoft.com/zh-cn/windows/desktop/gdi/device-contexts)  [GetDeviceCaps](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-getdevicecaps) |

## SelectObject函数的用法

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **SelectObject** 函数在 DC) (指定设备上下文中选择对象。 新 对象替换同一类型的上一个对象。 语法 C++复制  HGDIOBJ SelectObject(  [in] HDC hdc,  [in] HGDIOBJ h  ); 参数 [in] hdc  DC 的句柄。  [in] h  要选择的对象的句柄。 指定的对象必须已使用以下函数之一创建。  展开表   |  |  | | --- | --- | | **Object** | **函数** | | **Bitmap** | [CreateBitmap](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-createbitmap)、 [CreateBitmapIndirect](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-createbitmapindirect)、 [CreateCompatibleBitmap](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-createcompatiblebitmap)、 [CreateDIBitmap](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-createdibitmap)、 [CreateDIBSection](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-createdibsection)  位图只能选择到内存 DC 中。 不能将单个位图同时选入多个 DC。 | | **Brush** | [CreateBrushIndirect](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-createbrushindirect)、 [CreateDIBPatternBrush](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-createdibpatternbrush)、 [CreateDIBPatternBrushPt](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-createdibpatternbrushpt)、 [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) | | **字体** | [CreateFont](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-createfonta)、 [CreateFontIndirect](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-createfontindirecta) | | **笔** | [CreatePen](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-createpen)、 [CreatePenIndirect](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-createpenindirect) | | **区域** | [CombineRgn](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-combinergn)、 [CreateEllipticRgn](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-createellipticrgn)、 [CreateEllipticRgnIndirect](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-createellipticrgnindirect)、 [CreatePolygonRgn](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-createpolygonrgn)、 [CreateRectRgn](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-createrectrgn)、 [CreateRectRgnIndirect](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-createrectrgnindirect) |  返回值 如果所选对象不是区域且函数成功，则返回值是所替换对象的句柄。 如果所选对象是区域且函数成功，则返回值是以下值之一。  展开表   |  |  | | --- | --- | | **值** | **含义** | | SIMPLEREGION | 区域由单个矩形组成。 | | COMPLEXREGION | 区域由多个矩形组成。 | | NULLREGION | 区域为空。 |     如果发生错误，并且所选对象不是区域，则返回值为 **NULL**。 否则，它将HGDI\_ERROR。 注解 此函数返回指定类型的以前选择的对象。 应用程序在用新对象完成绘图后，应始终将新对象替换为原始的默认对象。  应用程序不能一次选择一个位图进入多个 DC。  **Icm：** 如果要选择的对象是画笔或笔，则执行颜色管理。 示例 有关示例，请参阅 [设置笔或画笔颜色](https://learn.microsoft.com/zh-cn/windows/desktop/gdi/setting-the-pen-or-brush-color)。 要求  |  |  | | --- | --- | | **标头** | wingdi.h (包括 Windows.h) | | **Library** | Gdi32.lib | | **DLL** | Gdi32.dll |  另请参阅 [CombineRgn](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-combinergn)  [CreateBitmap](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-createbitmap)  [CreateBitmapIndirect](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-createbitmapindirect)  [CreateBrushIndirect](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-createbrushindirect)  [CreateCompatibleBitmap](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-createcompatiblebitmap)  [CreateDIBPatternBrush](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-createdibpatternbrush)  [CreateDIBitmap](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-createdibitmap)  [CreateEllipticRgn](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-createellipticrgn)  [CreateEllipticRgnIndirect](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-createellipticrgnindirect)  [CreateFont](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-createfonta)  [CreateFontIndirect](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-createfontindirecta)  [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)  [CreatePen](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-createpen)  [CreatePenIndirect](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-createpenindirect)  [CreatePolygonRgn](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-createpolygonrgn)  [CreateRectRgn](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-createrectrgn)  [CreateRectRgnIndirect](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-createrectrgnindirect)  [CreateSolidBrush](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-createsolidbrush)  [设备上下文函数](https://learn.microsoft.com/zh-cn/windows/desktop/gdi/device-context-functions)  [设备上下文概述](https://learn.microsoft.com/zh-cn/windows/desktop/gdi/device-contexts)  [SelectClipRgn](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-selectcliprgn)  [SelectPalette](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-selectpalette) |

## BitBlt函数的用法

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **BitBlt** 函数执行与像素矩形相对应的颜色数据的位块传输，从指定的源设备上下文传输到目标设备上下文。 语法 C++  BOOL BitBlt(  [in] HDC hdc,  [in] int x,  [in] int y,  [in] int cx,  [in] int cy,  [in] HDC hdcSrc,  [in] int x1,  [in] int y1,  [in] DWORD rop  ); 参数 [in] hdc  目标设备上下文的句柄。  [in] x  目标矩形左上角的 x 坐标（以逻辑单位为单位）。  [in] y  目标矩形左上角的 y 坐标（以逻辑单位为单位）。  [in] cx  源矩形和目标矩形的宽度（以逻辑单位为单位）。  [in] cy  源矩形和目标矩形的高度（以逻辑单位为单位）。  [in] hdcSrc  源设备上下文的句柄。  [in] x1  源矩形左上角的 x 坐标（以逻辑单位为单位）。  [in] y1  源矩形左上角的 y 坐标（以逻辑单位为单位）。  [in] rop  光栅操作代码。 这些代码定义如何将源矩形的颜色数据与目标矩形的颜色数据相结合，以实现最终颜色。  以下列表显示了一些常见的光栅操作代码。  展开表   |  |  | | --- | --- | | **值** | **含义** | | **黑暗** | 使用与物理调色板中的索引 0 关联的颜色填充目标矩形。 （对于默认的物理调色板，该颜色为黑色。） | | **CAPTUREBLT** | 包括生成图像中窗口顶部分层的任何窗口。 默认情况下，图像仅包含你的窗口。 请注意，这通常无法用于打印设备上下文。 | | **DSTINVERT** | 反转目标矩形。 | | **MERGECOPY** | 使用布尔 AND 运算符将源矩形的颜色与 *hdcDest* 中当前选择的画笔合并。 | | **MERGEPAINT** | 使用布尔 OR 运算符将倒置源矩形的颜色与目标矩形的颜色合并。 | | **NOMIRRORBITMAP** | 防止位图镜像。 | | **NOTSRCCOPY** | 将反转源矩形复制到目标。 | | **NOTSRCERASE** | 使用布尔 OR 运算符组合源矩形和目标矩形的颜色，然后反转生成的颜色。 | | **PATCOPY** | 将 *hdcDest* 中当前选择的画笔复制到目标位图中。 | | **PATINVERT** | 使用布尔 XOR 运算符将 *hdcDest* 中当前选择的画笔的颜色与目标矩形的颜色组合在一起。 | | **PATPAINT** | 使用布尔 OR 运算符将 *hdcDest* 中当前选择的画笔的颜色与反转源矩形的颜色相结合。 此操作的结果通过使用布尔 OR 运算符与目标矩形的颜色相结合。 | | **SRCAND** | 使用布尔 AND 运算符组合源矩形和目标矩形的颜色。 | | **SRCCOPY** | 将源矩形直接复制到目标矩形。 | | **SRCERASE** | 使用布尔 AND 运算符将目标矩形的反转颜色与源矩形的颜色组合在一起。 | | **SRCINVERT** | 使用布尔 XOR 运算符组合源矩形和目标矩形的颜色。 | | **SRCPAINT** | 使用布尔 OR 运算符组合源矩形和目标矩形的颜色。 | | **白** | 使用与物理调色板中的索引 1 关联的颜色填充目标矩形。 （对于默认的物理调色板，该颜色为白色。） |  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。 要获得更多的错误信息，请调用 GetLastError。 注解 **BitBlt** 仅在目标 DC 上执行剪裁。  如果旋转或剪切转换在源设备上下文中生效， **BitBlt** 将返回错误。 如果源设备上下文中存在其他转换 (并且匹配转换在目标设备上下文) 中无效，则根据需要拉伸、压缩或旋转目标设备上下文中的矩形。  如果源和目标设备上下文的颜色格式不匹配， **BitBlt** 函数会将源颜色格式转换为与目标格式匹配。  记录增强型图元文件时，如果源设备上下文标识增强型图元文件设备上下文，则会发生错误。  并非所有设备都支持 **BitBlt** 函数。 有关详细信息，请参阅 [GetDeviceCaps](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-getdevicecaps) 函数中的RC\_BITBLT光栅功能条目以及以下函数： [MaskBlt](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-maskblt)、 [PlgBlt](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-plgblt) 和 [StretchBlt](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-stretchblt)。  如果源和目标设备上下文表示不同的设备，**BitBlt** 将返回错误。 若要在不同设备的 DC 之间传输数据，请通过调用 [GetDIBits](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-getdibits) 将内存位图转换为 DIB。 若要向第二台设备显示 DIB，请调用 [SetDIBits](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-setdibits) 或 [StretchDIBits](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-stretchdibits)。  **Icm：** 发生 blits 时，不执行颜色管理。 示例 下面的代码示例演示如何使用 **BitBlt**。  C++  if (!BitBlt(hdcMemDC,  0, 0,  rcClient.right - rcClient.left, rcClient.bottom - rcClient.top,  hdcWindow,  0, 0,  SRCCOPY))  {  MessageBox(hWnd, L"BitBlt has failed", L"Failed", MB\_OK);  goto done;  }  若要在上下文中查看此示例，请参阅 [捕获图像](https://learn.microsoft.com/zh-cn/windows/desktop/gdi/capturing-an-image)。 要求  |  |  | | --- | --- | | **标头** | wingdi.h (包括 Windows.h) | | **Library** | Gdi32.lib | | **DLL** | Gdi32.dll |  另请参阅 [位图函数](https://learn.microsoft.com/zh-cn/windows/desktop/gdi/bitmap-functions)  [位图概述](https://learn.microsoft.com/zh-cn/windows/desktop/gdi/bitmaps)  [GetDIBits](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-getdibits)  [GetDeviceCaps](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-getdevicecaps)  [MaskBlt](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-maskblt)  [PlgBlt](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-plgblt)  [SetDIBits](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-setdibits)  [StretchBlt](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-stretchblt)  [StretchDIBits](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-stretchdibits) |

## StretchBlt函数的用法：和BitBlt类似只不过它多了2个参数

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **StretchBlt** 函数将位图从源矩形复制到目标矩形，并在必要时拉伸或压缩位图以适应目标矩形的尺寸。 系统根据当前在目标设备上下文中设置的拉伸模式拉伸或压缩位图。 语法 C++  BOOL StretchBlt(  [in] HDC hdcDest,  [in] int xDest,  [in] int yDest,  [in] int wDest,  [in] int hDest,  [in] HDC hdcSrc,  [in] int xSrc,  [in] int ySrc,  [in] int wSrc,  [in] int hSrc,  [in] DWORD rop  ); 参数 [in] hdcDest  目标设备上下文的句柄。  [in] xDest  目标矩形左上角的 x 坐标（以逻辑单位为单位）。  [in] yDest  目标矩形左上角的 y 坐标（以逻辑单位为单位）。  [in] wDest  目标矩形的宽度（以逻辑单元表示）。  [in] hDest  目标矩形的高度（以逻辑单元表示）。  [in] hdcSrc  源设备上下文的句柄。  [in] xSrc  源矩形左上角的 x 坐标（以逻辑单位为单位）。  [in] ySrc  源矩形左上角的 y 坐标（以逻辑单位为单位）。  [in] wSrc  源矩形的宽度（以逻辑单元表示）。  [in] hSrc  源矩形的高度（以逻辑单元表示）。  [in] rop  要执行的光栅操作。 光栅操作代码定义系统如何在涉及画笔、源位图和目标位图的输出操作中组合颜色。  有关 IOPS) (常见光栅操作代码的列表，请参阅 [BitBlt](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-bitblt) 。 请注意，捕获BLT ROP 通常不能用于打印设备上下文。 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 **StretchBlt** 在内存中拉伸或压缩源位图，然后将结果复制到目标矩形。 此位图可以是兼容位图 (DDB) ，也可以是 [CreateDIBSection](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-createdibsection) 的输出。 在拉伸或压缩发生后，将合并图案或目标像素的颜色数据。  记录增强型图元文件时， (发生错误，如果源设备上下文标识增强型图元文件设备上下文，则函数返回 **FALSE**) 。  如果指定的光栅操作需要画笔，系统会使用当前在目标设备上下文中选择的画笔。  通过使用当前为目标设备上下文指定的转换来转换目标坐标;通过使用当前为源设备上下文指定的转换来转换源坐标。  如果源转换具有旋转或剪切，则会发生错误。  如果目标位图、源位图和图案位图的颜色格式不同， **StretchBlt** 会将源位图和图案位图转换为与目标位图匹配。  如果 **StretchBlt** 必须将单色位图转换为彩色位图，则会将白色位 (1) 设置为背景色，将黑色位设置为 (0) 前景色。 若要将颜色位图转换为单色位图，它将背景色匹配的像素设置为白色 (1) ，并将所有其他像素设置为黑色 (0) 。 在转换中将使用彩色设备上下文的前景色和背景色。  如果 *nWidthSrc 和 nWidthDest* 参数的符号或者 *nHeightSrc 和nHeightDest* 参数不同，**StretchBlt** 将创建位图的镜像图像。 如果 *nWidthSrc* 和 *nWidthDest* 具有不同的符号，则 函数会沿 x 轴创建位图的镜像图像。 如果 *nHeightSrc* 和 *nHeightDest* 具有不同的符号，则 函数将沿 y 轴创建位图的镜像图像。  并非所有设备都支持 **StretchBlt** 函数。 有关详细信息，请参阅 [GetDeviceCaps](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-getdevicecaps)。  **Icm：** 发生 blit 操作时，不执行颜色管理。  在多监视器系统中使用时， *hdcSrc* 和 *hdcDest* 必须引用同一设备，否则函数将失败。 若要在不同设备的 DC 之间传输数据，请通过调用 [GetDIBits](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-getdibits) 将内存位图转换为 DIB。 若要向第二台设备显示 DIB，请调用 [SetDIBits](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-setdibits) 或 [StretchDIBits](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-stretchdibits)。 示例 有关示例，请参阅 [缩放图像](https://learn.microsoft.com/zh-cn/windows/desktop/gdi/scaling-an-image)。 要求  |  |  | | --- | --- | | **目标平台** | Windows | | **标头** | wingdi.h (包括 Windows.h) | | **Library** | Gdi32.lib | | **DLL** | Gdi32.dll |  另请参阅 [BitBlt](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-bitblt)  [位图函数](https://learn.microsoft.com/zh-cn/windows/desktop/gdi/bitmap-functions)  [位图概述](https://learn.microsoft.com/zh-cn/windows/desktop/gdi/bitmaps)  [CreateDIBSection](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-createdibsection)  [GetDIBits](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-getdibits)  [GetDeviceCaps](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-getdevicecaps)  [MaskBlt](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-maskblt)  [PlgBlt](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-plgblt)  [SetDIBits](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-setdibits)  [SetStretchBltMode](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-setstretchbltmode)  [StretchDIBits](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-stretchdibits) |

## LoadImage函数的用法

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 加载图标、光标、动画光标或位图。 语法 C++复制  HANDLE LoadImageA(  [in, optional] HINSTANCE hInst,  [in] LPCSTR name,  [in] UINT type,  [in] int cx,  [in] int cy,  [in] UINT fuLoad  ); 参数 [in, optional] hInst  类型： **HINSTANCE**  包含要加载的图像的 DLL 或可执行文件 (.exe) 的模块的句柄。 有关详细信息，请参阅 [GetModuleHandle](https://learn.microsoft.com/zh-cn/windows/desktop/api/libloaderapi/nf-libloaderapi-getmodulehandlea)。 请注意，从 32 位 Windows 开始，实例句柄 (**HINSTANCE**) ，例如 [由 WinMain](https://learn.microsoft.com/zh-cn/windows/desktop/api/winbase/nf-winbase-winmain) 的系统函数调用公开的应用程序实例句柄，而模块句柄 (**HMODULE**) 也是如此。  若要 (图标、光标或位图文件) 加载预定义图像或独立资源，请将此参数设置为 **NULL**。  [in] name  类型： **LPCTSTR**  要加载的图像。  如果 *hInst* 参数为非 **NULL** 且 *fuLoad* 参数省略 **LR\_LOADFROMFILE**， *则 name* 指定 *hInst* 模块中的图像资源。  如果要按名称从模块加载图像资源， *则 name* 参数是指向包含映像资源名称的以 null 结尾的字符串的指针。  如果要从模块按序号加载图像资源，请使用 [MAKEINTRESOURCE](https://learn.microsoft.com/zh-cn/windows/desktop/api/winuser/nf-winuser-makeintresourcea) 宏将图像序号转换为可传递给 **LoadImage** 函数的形式。  如果 *hInst* 参数为 **NULL** 且 *fuLoad* 参数省略 **LR\_LOADFROMFILE** 值并包含 **LR\_SHARED**，则 *名称* 将指定要加载的预定义图像。  预定义的图像标识符在 中 Winuser.h 定义，并具有以下前缀：  展开表   | **前缀** | **含义** | | --- | --- | | **OBM\_** | OEM 位图。 使用 [MAKEINTRESOURCE](https://learn.microsoft.com/zh-cn/windows/desktop/api/winuser/nf-winuser-makeintresourcea) 宏传递这些。 | | **OIC\_** | OEM 图标。 使用 [MAKEINTRESOURCE](https://learn.microsoft.com/zh-cn/windows/desktop/api/winuser/nf-winuser-makeintresourcea) 宏传递这些。 | | **Ocr\_** | OEM 游标。 使用 [MAKEINTRESOURCE](https://learn.microsoft.com/zh-cn/windows/desktop/api/winuser/nf-winuser-makeintresourcea) 宏传递这些。 | | **IDI\_** | [标准图标](https://learn.microsoft.com/zh-cn/windows/win32/menurc/about-icons) | | **Idc\_** | [标准游标](https://learn.microsoft.com/zh-cn/windows/win32/menurc/about-cursors) |   若要将 OEM 图像标识符常量传递给 **LoadImage** 函数，请使用 [MAKEINTRESOURCE](https://learn.microsoft.com/zh-cn/windows/desktop/api/winuser/nf-winuser-makeintresourcea) 宏。 例如，若要加载**OCR\_NORMAL**游标，请将 作为 *name* 参数传递MAKEINTRESOURCE(OCR\_NORMAL)，**将 NULL** 作为 *hInst* 参数传递，**并将LR\_SHARED**作为标志之一传递给 *fuLoad* 参数。  如果 *hInst* 参数为 **NULL** 且 *fuLoad* 参数包含**LR\_LOADFROMFILE**值，*则 name* 是包含独立资源 (图标、光标或位图文件) 的文件的名称，例如 。 c:\myicon.ico  有关详细信息，请参阅下面的“备注”部分。  [in] type  类型： **UINT**  要加载的图像的类型。  此参数可能是以下值之一：  展开表   | **值** | **含义** | | --- | --- | | **IMAGE\_BITMAP** | 加载位图。 | | **IMAGE\_CURSOR** | 加载游标。 | | **IMAGE\_ICON** | 加载图标。 |   [in] cx  类型： **int**  图标或光标的宽度（以像素为单位）。 如果此参数为零且 *fuLoad* 参数 **为LR\_DEFAULTSIZE**，则函数使用 **SM\_CXICON** 或 **SM\_CXCURSOR** 系统指标值来设置宽度。 如果此参数为零且未使用 **LR\_DEFAULTSIZE** ，则函数使用实际资源宽度。  [in] cy  类型： **int**  图标或光标的高度（以像素为单位）。 如果此参数为零且 *fuLoad* 参数 **为LR\_DEFAULTSIZE**，则函数使用 **SM\_CYICON** 或 **SM\_CYCURSOR** 系统指标值来设置高度。 如果此参数为零且未使用 **LR\_DEFAULTSIZE** ，则函数使用实际资源高度。  [in] fuLoad  类型： **UINT**  此参数可使用以下一个或多个值。  展开表   |  |  | | --- | --- | | **值** | **含义** | | **LR\_CREATEDIBSECTION**  0x00002000 | 当 *uType* 参数指定 **IMAGE\_BITMAP**时，会导致函数返回 DIB 节位图而不是兼容的位图。 此标志可用于加载位图而不将其映射到显示设备的颜色。 | | **LR\_DEFAULTCOLOR**  0x00000000 | 默认标志;它不执行任何工作。 它的意思是“不 **LR\_MONOCHROME**”。 | | **LR\_DEFAULTSIZE**  0x00000040 | 如果 *cxDesired* 或 *cyDesired* 值设置为零，则使用游标或图标的系统指标值指定的宽度或高度。 如果未指定此标志，并且 *cxDesired* 和 *cyDesired* 设置为零，则函数将使用实际资源大小。 如果资源包含多个图像，则 函数使用第一个图像的大小。 | | **LR\_LOADFROMFILE**  0x00000010 | 从 *名称* (图标、光标或位图文件指定的文件) 加载独立图像。 | | **LR\_LOADMAP3DCOLORS**  0x00001000 | 在颜色表中搜索图像，并将以下灰色底纹替换为相应的三维颜色。   * Dk 灰色，RGB (128,128,128) 与 **COLOR\_3DSHADOW** * 灰色，RGB (192,192,192) ，带 **COLOR\_3DFACE** * Lt Gray，RGB (223,223,223) 与 **COLOR\_3DLIGHT**   如果要加载颜色深度大于 8bpp 的位图，请不要使用此选项。 | | **LR\_LOADTRANSPARENT**  0x00000020 | 检索图像中第一个像素的颜色值，并将颜色表中的相应条目替换为默认窗口颜色 (**COLOR\_WINDOW**) 。 图像中使用该条目的所有像素都将成为默认的窗口颜色。 此值仅适用于具有相应颜色表的图像。  如果要加载颜色深度大于 8bpp 的位图，请不要使用此选项。  如果 *fuLoad* 同时包含 **LR\_LOADTRANSPARENT** 值和 **LR\_LOADMAP3DCOLORS** 值， **LR\_LOADTRANSPARENT** 优先。 但是，颜色表条目将替换为 **COLOR\_3DFACE** 而不是 **COLOR\_WINDOW**。 | | **LR\_MONOCHROME**  0x00000001 | 加载黑白图像。 | | **LR\_SHARED**  0x00008000 | 如果多次加载映像，则共享映像句柄。 如果未设置 **LR\_SHARED** ，则对同一资源的第二次 **LoadImage** 调用将再次加载映像并返回不同的句柄。  使用此标志时，系统会在不再需要资源时销毁该资源。  对于具有非标准大小、加载后可能会更改或从文件加载的图像，请勿使用 **LR\_SHARED** 。  加载系统图标或光标时，必须使用 **LR\_SHARED** 否则函数将无法加载资源。  无论请求的大小如何，此函数都会查找缓存中具有请求的资源名称的第一个映像。 | | **LR\_VGACOLOR**  0x00000080 | 使用真正的 VGA 颜色。 |  返回值 类型： **HANDLE**  如果函数成功，则返回值是新加载的图像的句柄。  如果函数失败，则返回值为 NULL。 要获得更多的错误信息，请调用 GetLastError。 注解 如果 [IS\_INTRESOURCE](https://learn.microsoft.com/zh-cn/windows/desktop/api/winuser/nf-winuser-is_intresource) (*名称*) 为 **TRUE**，则 *name* 指定给定资源的整数标识符。 否则，它是指向以 null 结尾的字符串的指针。 如果字符串的第一个字符是井号 (#) ，则其余字符表示一个指定资源的整数标识符的十进制数。 例如，字符串“#258”表示标识符 258。  使用完未指定 **LR\_SHARED** 标志的情况下加载的位图、光标或图标后，可以通过调用下表中的函数之一释放其关联的内存。  展开表   |  |  | | --- | --- | | **资源** | **Release 函数** | | Bitmap | [DeleteObject](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-deleteobject) | | 游标 | [DestroyCursor](https://learn.microsoft.com/zh-cn/windows/desktop/api/winuser/nf-winuser-destroycursor) | | 图标 | [DestroyIcon](https://learn.microsoft.com/zh-cn/windows/desktop/api/winuser/nf-winuser-destroyicon) |     当创建这些资源的进程终止时，系统会自动删除这些资源;但是，调用相应的函数可节省内存并减小进程的工作集的大小。 示例 有关示例，请参阅 [使用窗口类](https://learn.microsoft.com/zh-cn/windows/desktop/winmsg/using-window-classes)。  **备注**  winuser.h 标头将 LoadImage 定义为别名，该别名根据 UNICODE 预处理器常量的定义自动选择此函数的 ANSI 或 Unicode 版本。 将非特定编码别名与非非特定编码的代码混合使用可能会导致不匹配，从而导致编译或运行时错误。 有关详细信息，请参阅 [**函数原型的约定**](https://learn.microsoft.com/zh-cn/windows/win32/intl/conventions-for-function-prototypes)。 要求  |  |  | | --- | --- | | **标头** | winuser.h (包括 Windows.h) | | **Library** | User32.lib | | **DLL** | User32.dll | | **API 集** | windows 8 中引入的 ext-ms-win-ntuser-gui-l1-1-0 () |  请参阅 **概念性**  [CopyImage](https://learn.microsoft.com/zh-cn/windows/desktop/api/winuser/nf-winuser-copyimage)  [GetSystemMetrics](https://learn.microsoft.com/zh-cn/windows/desktop/api/winuser/nf-winuser-getsystemmetrics)  [LoadBitmap](https://learn.microsoft.com/zh-cn/windows/desktop/api/winuser/nf-winuser-loadbitmapa)  [LoadCursor](https://learn.microsoft.com/zh-cn/windows/desktop/api/winuser/nf-winuser-loadcursora)  [LoadIcon](https://learn.microsoft.com/zh-cn/windows/desktop/api/winuser/nf-winuser-loadicona)  **其他资源**  **引用**  [资源](https://learn.microsoft.com/zh-cn/windows/desktop/menurc/resources) |

## DrawFrameControl函数的用法

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **DrawFrameControl** 函数绘制指定类型和样式的帧控件。 语法 C++复制  BOOL DrawFrameControl(  [in] HDC hdc,  [in] LPRECT lprc,  [in] UINT uType,  [in] UINT uState  ); 参数 [in] hdc  要在其中绘制控件的窗口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，该结构包含框架控件的边框的逻辑坐标。  [in] uType  要绘制的帧控件的类型。 此参数的取值可为下列值之一：  展开表   |  |  | | --- | --- | | **值** | **含义** | | **DFC\_BUTTON** | “标准”按钮 | | **DFC\_CAPTION** | 标题栏 | | **DFC\_MENU** | 菜单栏 | | **DFC\_POPUPMENU** | 弹出菜单项。 | | **DFC\_SCROLL** | 滚动条 |   [in] uState  帧控件的初始状态。 如果 *uType* 为DFC\_BUTTON， *则 uState* 可以是以下值之一。  展开表   |  |  | | --- | --- | | **值** | **含义** | | **DFCS\_BUTTON3STATE** | 三态按钮 | | **DFCS\_BUTTONCHECK** | 复选框 | | **DFCS\_BUTTONPUSH** | 下压按钮 | | **DFCS\_BUTTONRADIO** | 单选按钮 | | **DFCS\_BUTTONRADIOIMAGE** | 单选按钮的图像（非正方形需要图像） | | **DFCS\_BUTTONRADIOMASK** | 单选按钮的掩码（非正方形需要掩码） |     如果 *uType* 为DFC\_CAPTION， *则 uState* 可以是以下值之一。  展开表   |  |  | | --- | --- | | **值** | **含义** | | **DFCS\_CAPTIONCLOSE** | **“关闭** ”按钮 | | **DFCS\_CAPTIONHELP** | **“帮助** ”按钮 | | **DFCS\_CAPTIONMAX** | **“最大化** ”按钮 | | **DFCS\_CAPTIONMIN** | **最小化** 按钮 | | **DFCS\_CAPTIONRESTORE** | **“还原”** 按钮 |     如果 *uType* DFC\_MENU， *则 uState* 可以是以下值之一。  展开表   |  |  | | --- | --- | | **值** | **含义** | | **DFCS\_MENUARROW** | 子菜单箭头 | | **DFCS\_MENUARROWRIGHT** | 向左方向的子菜单箭头。 这用于从右到左语言（如阿拉伯语或希伯来语）使用的从右到左级联菜单。 | | **DFCS\_MENUBULLET** | 子弹 | | **DFCS\_MENUCHECK** | 选中标记 |     如果 *uType* 为DFC\_SCROLL， *则 uState* 可以是以下值之一。  展开表   |  |  | | --- | --- | | **值** | **含义** | | **DFCS\_SCROLLCOMBOBOX** | 组合框滚动条 | | **DFCS\_SCROLLDOWN** | 滚动条的向下箭头 | | **DFCS\_SCROLLLEFT** | 滚动条的向左箭头 | | **DFCS\_SCROLLRIGHT** | 滚动条的向右箭头 | | **DFCS\_SCROLLSIZEGRIP** | 窗口右下角的大小手柄 | | **DFCS\_SCROLLSIZEGRIPRIGHT** | 窗口左下角的大小手柄。 这与从右到左的语言（如阿拉伯语或希伯来语）一起使用。 | | **DFCS\_SCROLLUP** | 滚动条的向上箭头 |     以下样式可用于调整按钮的边框。  展开表   |  |  | | --- | --- | | **值** | **含义** | | **DFCS\_ADJUSTRECT** | 调整边框以排除按钮的周围边缘。 |     以下一个或多个值可用于设置要绘制的控件的状态。  展开表   |  |  | | --- | --- | | **值** | **含义** | | **DFCS\_CHECKED** | 已选中按钮。 | | **DFCS\_FLAT** | 按钮有一个平面边框。 | | **DFCS\_HOT** | 按钮是热跟踪的。 | | **DFCS\_INACTIVE** | 按钮处于非活动状态， (灰色) 。 | | **DFCS\_MONO** | 按钮具有单色边框。 | | **DFCS\_PUSHED** | 按钮已按下。 | | **DFCS\_TRANSPARENT** | 背景保持不变。 此标志只能与DFCS\_MENUARROWUP或DFCS\_MENUARROWDOWN结合使用。 |  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 如果 *uType* 为DFC\_MENU或DFC\_BUTTON且 *uState* 未DFCS\_BUTTONPUSH，则帧控件为黑白蒙板 (即白色背景上的黑色框架控件) 。 在这种情况下，应用程序必须将句柄传递给位图内存设备控件。 然后，应用程序可以使用关联的位图作为 [MaskBlt](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-maskblt) 函数的 *hbmMask* 参数，或者可以使用 RP（如 SRCAND 和 SRCINVERT）将设备上下文用作 [BitBlt](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-bitblt) 函数的参数。 DPI 虚拟化 此 API 不参与 DPI 虚拟化。 给定的输入始终以物理像素为单位，与调用上下文无关。 要求  |  |  | | --- | --- | | **标头** | winuser.h (包括 Windows.h) | | **Library** | User32.lib | | **DLL** | User32.dll | | **API 集** | ext-ms-win-ntuser-draw-l1-1-2 (在 Windows 10 版本 10.0.10240 中引入) |  另请参阅 [绘制和绘制函数](https://learn.microsoft.com/zh-cn/windows/desktop/gdi/painting-and-drawing-functions)  [绘画和绘图概述](https://learn.microsoft.com/zh-cn/windows/desktop/gdi/painting-and-drawing)  [矩形](https://learn.microsoft.com/zh-cn/windows/desktop/api/windef/ns-windef-rect) |

# 演练

## 1.新建一个win32项目，取名Lesson47-bitmap-demo

|  |
| --- |
|  |

## 2.然后我们把3个位图图片放入cpp源文件目录下面

|  |
| --- |
|  |

## 3.在这个文件夹里面新建一个res文件夹，然后把位图移动到里面

|  |
| --- |
|  |

## 4.然后右键点击资源文件夹-》添加资源=》bitmap-》导入，进入位图所在的文件夹

|  |
| --- |
|  |

## 5.选中所有位图，点击打开就可以导入位图

|  |
| --- |
|  |

### 注意：需要点击保存按钮保存导入的资源

## 6.先定义3个位图句柄来保存加载的位图

|  |
| --- |
|  |

## 7.我们需要在MyRegisterClass函数里面给窗口开启接受左键双击的功能

|  |
| --- |
|  |

## 8.然后我们添加两个消息处理代码分别是WM\_LBUTTONDBLCLK和WM\_RBUTTONDOWN，需要在这里修改两个保证变量的值并且发送WM\_PAINT消息和调用InvalidateRect函数使得地区客户区无效这样子就好触发WM\_PAINT消息处理代码的执行，注意需要SendMessage和InvalidateRect函数一起使用，只是用SendMessage的话，需要改变窗口的大小才有效

|  |
| --- |
|  |

## 9.然后我们来响应左键双击事件和右击事件

|  |
| --- |
|  |

#### 注意：不要在WM\_SIZE消息处理代码里面发送 WM\_PAINT消息和调用InvalidateRect函数，会导致窗口显示不了图片。

### 运行程序，双击设备左键，效果如下

|  |
| --- |
|  |

### 按下鼠标右击，效果如下

|  |
| --- |
|  |

### 改变窗口大小，效果如下，是空白因为此时显示默认位图的函数还没有完成

|  |
| --- |
|  |

## 10.我们需要添加一个BITMAP结构体变量用来保存位图的宽高和两个表示宽度和高度的变量还需要一个兼容DC句柄变量

|  |
| --- |
|  |

## 11，回到WM\_PAINT显示处理代码，我们来加载位图

|  |
| --- |
| case WM\_PAINT:  //加载位图  bm1 = LoadBitmap(hInst,MAKEINTRESOURCE(IDB\_BITMAP1));  bm2 = LoadBitmap(hInst,MAKEINTRESOURCE(IDB\_BITMAP2));  bm3 = LoadBitmap(hInst,MAKEINTRESOURCE(IDB\_BITMAP3));  //保存位图1的大小  GetObject(bm1,sizeof(BITMAP),&bitmap);  bWidth = bitmap.bmWidth;  bHeight = bitmap.bmHeight;  //开始绘图  hdc = BeginPaint(hWnd, &ps);  //显示位图需要创建窗口dc的兼容DC  hMemDC = CreateCompatibleDC(hdc);  //先把位图选进兼容DC，  SelectObject(hMemDC,bm1);  ////3.显示位图，把窗口dc作为目标dc，兼容dc作为源DC  BitBlt(hdc,350,100,bWidth,bHeight,hMemDC,0,0,SRCCOPY);  //ShowInitBitmap(hdc,bm1);  GetClientRect(hWnd,&rc);    if(isLeft)//鼠标左键双击的时候用这个位图画刷填充窗口  {  //MessageBox(hWnd,\_T("PAINT"),\_T("TEST"),0);  brush2 = CreatePatternBrush(bm2);  //FillRect(hdc,&rc,CreateSolidBrush(RGB(255,0,255)));  FillRect(hdc,&rc,brush2);  }  else if(isRight)//点击鼠标右击时用这个位图画刷填充窗口  {  brush3 = CreatePatternBrush(bm3);  FillRect(hdc,&rc,brush3);  }    EndPaint(hWnd, &ps);  break; |

### 12.也可以使用StretchBlt函数，它可以拉伸位图来适应窗口

|  |
| --- |
| case WM\_PAINT:  GetClientRect(hWnd,&rc);  //加载位图  bm1 = LoadBitmap(hInst,MAKEINTRESOURCE(IDB\_BITMAP1));  bm2 = LoadBitmap(hInst,MAKEINTRESOURCE(IDB\_BITMAP2));  bm3 = LoadBitmap(hInst,MAKEINTRESOURCE(IDB\_BITMAP3));  //保存位图1的大小  GetObject(bm1,sizeof(BITMAP),&bitmap);  bWidth = bitmap.bmWidth;  bHeight = bitmap.bmHeight;  //开始绘图  hdc = BeginPaint(hWnd, &ps);  //显示位图需要创建窗口dc的兼容DC  hMemDC = CreateCompatibleDC(hdc);  //先把位图选进兼容DC，  SelectObject(hMemDC,bm1);  ////3.显示位图，把窗口dc作为目标dc，兼容dc作为源DC  //BitBlt(hdc,350,100,bWidth,bHeight,hMemDC,0,0,SRCCOPY);  StretchBlt(hdc,0,0,rc.left+rc.right,rc.top+rc.bottom,hMemDC,0,0,bWidth,bHeight,SRCCOPY);  //ShowInitBitmap(hdc,bm1);    if(isLeft)//鼠标左键双击的时候用这个位图画刷填充窗口  {  //MessageBox(hWnd,\_T("PAINT"),\_T("TEST"),0);  brush2 = CreatePatternBrush(bm2);  //FillRect(hdc,&rc,CreateSolidBrush(RGB(255,0,255)));  FillRect(hdc,&rc,brush2);  }  else if(isRight)//点击鼠标右击时用这个位图画刷填充窗口  {  brush3 = CreatePatternBrush(bm3);  FillRect(hdc,&rc,brush3);  }    EndPaint(hWnd, &ps);  break; |

### 注意：不要随便去响应WM\_CREATE消息，有时候，它会引起WM\_PAINT的消息处理代码失效

# 这一节发学习到此为止，完整代码如下

## Lesson47-bitmap-demo.cpp

|  |
| --- |
| // Lesson47-bitmap-demo.cpp : 定义应用程序的入口点。  //  #include "stdafx.h"  #include "Lesson47-bitmap-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\_LESSON47BITMAPDEMO, 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\_LESSON47BITMAPDEMO));  // 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|CS\_DBLCLKS; //开启接受左键双击功能  wcex.lpfnWndProc = WndProc;  wcex.cbClsExtra = 0;  wcex.cbWndExtra = 0;  wcex.hInstance = hInstance;  wcex.hIcon = LoadIcon(hInstance, MAKEINTRESOURCE(IDI\_LESSON47BITMAPDEMO));  wcex.hCursor = LoadCursor(NULL, IDC\_ARROW);  wcex.hbrBackground = (HBRUSH)(COLOR\_WINDOW+1);  wcex.lpszMenuName = MAKEINTRESOURCE(IDC\_LESSON47BITMAPDEMO);  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,hMemDC;  RECT rc;  static HBITMAP bm1,bm2,bm3;  static HBRUSH brush1,brush2,brush3;  static BOOL isLeft = FALSE;  static BOOL isRight = FALSE;  static BITMAP bitmap;  static int bWidth,bHeight;//位图的宽度和高度    switch (message)  {    case WM\_LBUTTONDBLCLK://双击鼠标左键需要发送窗口重绘消息和使得地区客户区无效，这样子就会触发WM\_PAINT消息处理代码  //MessageBox(hWnd,\_T("Mouse Clicked"),\_T("TEST"),0);  isRight = FALSE;//双击鼠标左键后需要把是否右击设置为否  isLeft = TRUE;  SendMessage(hWnd,WM\_PAINT,0,0);  InvalidateRect(hWnd,NULL,TRUE);  break;  case WM\_RBUTTONDOWN://鼠标右键点击需要发送窗口重绘消息和使得地区客户区无效，这样子就会触发WM\_PAINT消息处理代码  isLeft = FALSE;//鼠标右击后需要把是否是鼠标双击设置为否  isRight = TRUE;  SendMessage(hWnd,WM\_PAINT,0,0);  InvalidateRect(hWnd,NULL,TRUE);  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:  GetClientRect(hWnd,&rc);  //加载位图  bm1 = LoadBitmap(hInst,MAKEINTRESOURCE(IDB\_BITMAP1));  bm2 = LoadBitmap(hInst,MAKEINTRESOURCE(IDB\_BITMAP2));  bm3 = LoadBitmap(hInst,MAKEINTRESOURCE(IDB\_BITMAP3));  //保存位图1的大小  GetObject(bm1,sizeof(BITMAP),&bitmap);  bWidth = bitmap.bmWidth;  bHeight = bitmap.bmHeight;  //开始绘图  hdc = BeginPaint(hWnd, &ps);  //显示位图需要创建窗口dc的兼容DC  hMemDC = CreateCompatibleDC(hdc);  //先把位图选进兼容DC，  SelectObject(hMemDC,bm1);  ////3.显示位图，把窗口dc作为目标dc，兼容dc作为源DC  //BitBlt(hdc,350,100,bWidth,bHeight,hMemDC,0,0,SRCCOPY);  StretchBlt(hdc,0,0,rc.left+rc.right,rc.top+rc.bottom,hMemDC,0,0,bWidth,bHeight,SRCCOPY);  //ShowInitBitmap(hdc,bm1);    if(isLeft)//鼠标左键双击的时候用这个位图画刷填充窗口  {  //MessageBox(hWnd,\_T("PAINT"),\_T("TEST"),0);  brush2 = CreatePatternBrush(bm2);  //FillRect(hdc,&rc,CreateSolidBrush(RGB(255,0,255)));  FillRect(hdc,&rc,brush2);  }  else if(isRight)//点击鼠标右击时用这个位图画刷填充窗口  {  brush3 = CreatePatternBrush(bm3);  FillRect(hdc,&rc,brush3);  }    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;  } |