## WinAPI: InflateRect - 改变矩形大小

```
//声明:
InflateRect(
  var lprc: TRect; {要修改的矩形}
  dx, dy: Integer  {变化值}
): BOOL;
```

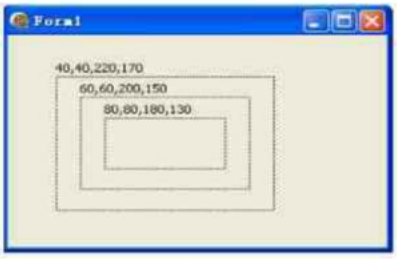
---

```
//举例:
procedure TForm1.FormPaint(Sender: TObject);
var
  R: TRect;
const
  s = '%d,%d,%d,%d';
begin
  R := Rect(60, 60, 200, 150);

  DrawFocusRect(Canvas.Handle, R);
  Canvas.TextOut(R.Left, R.Top-15, Format(s,[R.Left,R.Top,R.Right,R.Bottom]));

  InflateRect(R, 20, 20);
  DrawFocusRect(Canvas.Handle, R);
  Canvas.TextOut(R.Left, R.Top-15, Format(s,[R.Left,R.Top,R.Right,R.Bottom]));

  InflateRect(R, -40, -40);
  DrawFocusRect(Canvas.Handle, R);
  Canvas.TextOut(R.Left, R.Top-15, Format(s,[R.Left,R.Top,R.Right,R.Bottom]));
end;
```

---



```
//效果图:
```



**WinAPI: FlashWindow** - 闪烁窗口

```
//声明:
FlashWindow(
  hWnd: HWND;   {窗口句柄}
  bInvert: BOOL {设为 True 才会闪烁}
): BOOL;
```

---

```
//举例:
begin
  FlashWindow(Handle, True);
end;
```

---

**WinAPI: GetActiveWindow** - 获取当前活动窗口的句柄

```
//声明:
  GetActiveWindow: HWND; {无参数; 返回当前活动窗口的句柄}
```

---

```
//举例:
var
  h: HWND;
begin
  h := GetActiveWindow;
  FlashWindow(h, True);
end;
```

---

**WinAPI: GetFocus** - 获取当前拥有焦点的窗口的句柄

```
//声明:
GetFocus: HWND; {无参数; 返回当前拥有焦点窗口的句柄}
```



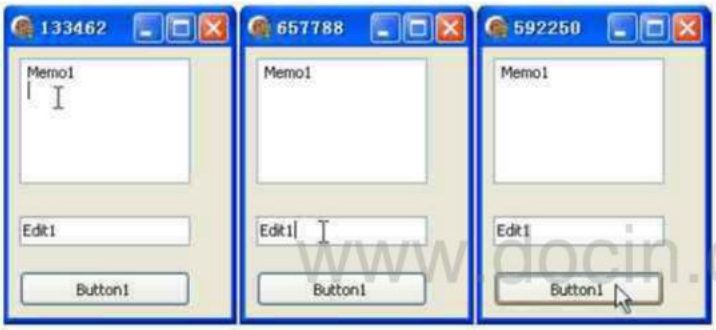
---

```
//举例:
unit Unit1;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
```

```
  Dialogs, StdCtrls, ExtCtrls;
type
  TForm1 = class(TForm)
    Button1: TButton;
    Memo1: TMemo;
    Edit1: TEdit;
    Timer1: TTimer;
    procedure Timer1Timer(Sender: TObject);
  end;
var
  Form1: TForm1;
implementation
{$R *.dfm}
procedure TForm1.Timer1Timer(Sender: TObject);
var
  h: HWND;
begin
  h := GetFocus;
  Text := IntToStr(h);
end;
end.
```

---

//效果图：






---

**WinAPI: GetParent** - 获取指定窗口的父窗口句柄

```
//声明：
GetParent(
  hWnd: HWND {窗口句柄}
```

```
): HWND;      {返回父窗口句柄}
```

```
//举例:
procedure TForm1.Button1Click(Sender: TObject);
var
  h1,h2: HWND;
begin
  h1 := GetParent(Button1.Handle);
  h2 := Panel1.Handle;
  ShowMessage(IntToStr(h1)); {590862}
  ShowMessage(IntToStr(h2)); {590862}

  h1 := GetParent(Panel1.Handle);
  h2 := Self.Handle;
  ShowMessage(IntToStr(h1)); {459824}
  ShowMessage(IntToStr(h2)); {459824}

  ShowMessage(IntToStr(GetParent(Handle))); {0}
end;
```

```
//效果图:
```




**WinAPI: GetParent** - 判断两个窗口是不是父子关系

```
//声明:
IsChild(
  hWndParent, hWnd: HWND {参数是两个窗口句柄, 父窗口在前}
): BOOL;
```

## WinAPI: IsIconic、IsZoomed - 分别判断窗口是否已最小化、最大化

```
//声明:
IsIconic(
  hWnd: HWND {窗口句柄}
): BOOL;

IsZoomed(
  hWnd: HWND {窗口句柄}
): BOOL;
```

---

## WinAPI: MoveWindow - 改变窗口的位置与大小

```
//声明:
MoveWindow(
  hWnd: HWND;              {窗口句柄}
  X, Y: Integer;           {位置}
  nWidth, nHeight: Integer; {大小}
  bRepaint: BOOL           {是否重绘}
): BOOL;
```

---

```
//举例:
procedure TForm1.Button1Click(Sender: TObject);
begin
  MoveWindow(Button1.Handle, 0, 0, ClientWidth, ClientHeight, True);
end;
```

---

//效果图:



**WinAPI: SetWindowPos** - 改变窗口的位置与状态

```
//声明:
SetWindowPos(
  hWnd: HWND;           {窗口句柄}
  hWndInsertAfter: HWND; {窗口的 Z 顺序}
  X, Y: Integer;        {位置}
  cx, cy: Integer;      {大小}
  uFlags: UINT          {选项}
): BOOL;
//hWndInsertAfter 参数可选值:
HWND_TOP      = 0;        {在前面}
HWND_BOTTOM   = 1;        {在后面}
HWND_TOPMOST  = HWND(-1); {在前面, 位于任何顶部窗口的前面}
HWND_NOTOPMOST = HWND(-2); {在前面, 位于其他顶部窗口的后面}
//uFlags 参数可选值:
SWP_NOSIZE        = 1;    {忽略 cx、cy, 保持大小}
SWP_NOMOVE        = 2;    {忽略 X、Y, 不改变位置}
SWP_NOZORDER      = 4;    {忽略 hWndInsertAfter, 保持 Z 顺序}
SWP_NOREDRAW      = 8;    {不重绘}
SWP_NOACTIVATE    = $10;  {不激活}
SWP_FRAMECHANGED  = $20;  {强制发送 WM_NCCALCSIZE 消息, 一般只是在改变大小时才发送此消息}
SWP_SHOWWINDOW    = $40;  {显示窗口}
SWP_HIDEWINDOW    = $80;  {隐藏窗口}
SWP_NOCOPYBITS    = $100; {丢弃客户区}
SWP_NOOWNERZORDER = $200; {忽略 hWndInsertAfter, 不改变 Z 序列的所有者}
```

```
SWP_NOSENDCHANGING = $400; {不发出 WM_WINDOWPOSCHANGING 消息}
SWP_DRAWFRAME      = SWP_FRAMECHANGED; {画边框}
SWP_NOREPOSITION   = SWP_NOOWNERZORDER;{}
SWP_DEFERERASE     = $2000;          {防止产生 WM_SYNCPAINT 消息}
SWP_ASYNCWINDOWPOS = $4000;          {若调用进程不拥有窗口，系统会向拥有窗口的线程发出需求}
```

---

```
//举例:
procedure TForm1.Button1Click(Sender: TObject);
begin
  SetWindowPos(Handle, HWND_TOPMOST, 0,0, 100,200, SWP_SHOWWINDOW);
end;
```

---

**WinAPI: WindowFromPoint- 获取指定点所在窗口的句柄**

```
//声明:
WindowFromPoint(Point: TPoint): HWND;
```

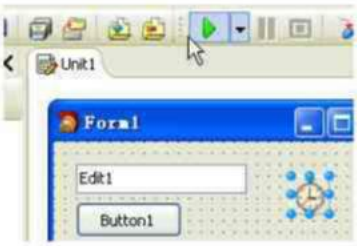
---

```
//举例:
unit Unit1;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, ExtCtrls;
type
  TForm1 = class(TForm)
    Button1: TButton;
    Timer1: TTimer;
    procedure Timer1Timer(Sender: TObject);
  end;
var
  Form1: TForm1;
implementation
{$R *.dfm}
var
  h: HWND;
```

```
procedure TForm1.Timer1Timer(Sender: TObject);
var
  pt: TPoint;
  arr: array[0..254] of Char;
begin
  if GetCursorPos(pt) then              {如果能获取点}
  begin
    h := WindowFromPoint(pt);           {返回句柄}
    GetClassName(h, arr, Length(arr)); {获取该句柄窗口的类名}
    Text := arr;                        {显示在标题}
  end;
end;
end.
```

---

//效果图:




---

## WinAPI: GetWindowRect、GetClientRect - 获取窗口的外部与内部矩形

提示:

1、其实用 Delphi 内部同类函数很方便的, 但系统函数是全局的;

2、使用 GetClientRect 时, 一般要 Windows.GetClientRect, 因为 TForm 的父类有同名函数.



---

//声明:

```
{获取窗口外部矩形(相对于屏幕)}
GetWindowRect(
  hWnd: HWND;       {窗口句柄}
  var lpRect: TRect {用于返回的矩形指针}
```

```
): BOOL;
{获取窗口内部矩形}
GetClientRect(
  hWnd: HWND;       {窗口句柄}
  var lpRect: TRect {用于返回的矩形指针}
): BOOL;
```

---

```
//举例:
unit Unit1;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, ExtCtrls;
type
  TForm1 = class(TForm)
    Label1: TLabel;
    Label2: TLabel;
    procedure FormShow(Sender: TObject);
  end;
var
  Form1: TForm1;
implementation
{$R *.dfm}
procedure TForm1.FormShow(Sender: TObject);
var
  r: TRect;
begin
  GetWindowRect(Handle, r);
  Label1.Caption := Format('%d,%d,%d,%d', [r.Left,r.Top,r.Right,r.Bottom]);
  Windows.GetClientRect(Handle, r);
  Label2.Caption := Format('%d,%d,%d,%d', [r.Left,r.Top,r.Right,r.Bottom]);
end;
end.
//效果图:
```



---


**WinApi: GetParent、SetParent、MoveWindow - 获取、指定父窗口和移动窗口**

提示: SetParent 应该 Windows.SetParent, 因为 TForm 的父类有同名方法.

```
//声明:
{获取父窗口句柄}
GetParent(hWnd: HWND): HWND;
{指定父窗口}
SetParent(
  hWndChild: HWND;    {子句柄}
  hWndNewParent: HWND {父句柄}
): HWND;              {成功返回原父窗口句柄; 失败返回 0}
{移动窗口}
MoveWindow(
  hWnd: HWND;              {窗口句柄}
  X, Y: Integer;           {位置}
  nWidth, nHeight: Integer; {大小}
  bRepaint: BOOL           {True 表示刷新; False 表示不刷新}
): BOOL;
```



```
//举例:
unit Unit1;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, ExtCtrls;
```

```
type
  TForm1 = class(TForm)
    Edit1: TEdit;
    Button1: TButton;
    procedure Button1Click(Sender: TObject);
  end;
var
  Form1: TForm1;
implementation
{$R *.dfm}
procedure TForm1.Button1Click(Sender: TObject);
begin
  if GetParent(Edit1.Handle)=Handle then
  begin
    Windows.SetParent(Edit1.Handle, Button1.Handle);
    MoveWindow(Edit1.Handle, 0,0, Edit1.Width, Edit1.Height, True);
  end else begin
    Windows.SetParent(Edit1.Handle, Self.Handle);
    MoveWindow(Edit1.Handle, 0,0, Edit1.Width, Edit1.Height, True);
  end;
end;
end.
```

---

//效果图: