1. Win32导出函数

1.1新建Win32 DLL工程，已自动生成cpp文件，但没有h文件，添加h头文件，在h头文件中添加：

#include "stdafx.h"

//导出函数

方法一：

extern "C" float \_declspec(dllexport)calculate(float x,float y,int type);

方法二：

定义DEF文件，并在DEF文件中输入以下代码：

LIBRARY "WinDLL"

EXPORTS

calculate @1

**不需要写头文件**

1.2 在cpp文件中正常实现函数：

#include "StdAfx.h"

#include "WinDLL.h"

float calculate(float x,float y,int type)

{

……

}

1.3 DLL的运用：

1.3.1、新建工程，将编译成功的DLL文件复制到工程目录下。

1.3.2、在cpp文件中设置lib文件路径，并导出函数：

方法一：

#pragma comment(lib,"G:\\其它\\VC驿站\\WinDLL\\Debug\\WinDLL.lib")

extern "C" float \_declspec(dllimport)calculate(float x,float y,int type);

方法二：

#pragma comment(lib,"G:\\其它\\VC驿站\\WinDLL\\Debug\\WinDLL.lib")

extern float calculate(float x,float y,int type);

注意这里不需要：“extern "C" float \_declspec(dllimport)”

这样，就可以在该工程中使用DLL中的函数calculate。“#pragma comment……”这句代码可以免去在在工程属性中设置附加依赖项。

1. win32导出类

2.1 在h文件中声明类，格式类似于：

class \_declspec(dllexport) 类名

{

……

}

注意，这里没有extern “C”

然后在cpp文件中正常实现类。

2.2 类的使用

新建工程，并复制DLL工程中的h文件到到新工程的h文件中，把dllexport换成dllimport。即:

class \_declspec(dllimport) 类名

{

……

}

在文件顶端指定lib文件位置：

#pragma comment(lib,"G:\\其它\\VC驿站\\WinDLL\\Debug\\WinDLL.lib")

实际上，无论是导出类还是导出函数，如果是\_declspec这种形式导出的，都可以把DLL工程中的头文件复制过去，然后把dllexport换成dllimport，再指定lib文件位置后便可正常使用。

1. 以下给出一个实例，实例中既导出了一个函数，也导出了一个类：

DLL工程：

CPP文件：

// WinDLL.cpp : 定义DLL 应用程序的导出函数

#include "StdAfx.h"

#include "WinDLL.h"

//导出函数

float calculate(float x,float y,int type)

{

switch(type)

{

头文件：

#include "stdafx.h"

//导出函数

extern "C" float \_declspec(dllexport)calculate(float x,float y,int type);

//导出类

class \_declspec(dllexport) CColumn

{

public:

CColumn();

CColumn(double r,double h);

double GetArea();

double GetVolum();

double GetRadial();

double GetHeight();

void SetRadial(double r);

void SetHeight(double h);

protected:

private:

double radial;

double height;

};

case 1:

return x+y;

break;

case 2:

return x-y;

break;

case 3:

return x\*y;

break;

case 4:

if (y!=0)

return x/y;

else

return 0;

break;

}

}

//类实现

CColumn::CColumn()

{

radial=0;

height=0;

}

CColumn::CColumn(double r,double h)

{

radial=r;

height=h;

}

double CColumn::GetArea()

{

return radial\*radial\*3.1415926;

}

double CColumn::GetVolum()

{

return radial\*radial\*3.1415926\*height;

}

void CColumn::SetRadial(double r)

{

radial=r;

}

void CColumn::SetHeight(double h)

{

height=h;

}

double CColumn::GetRadial()

{

return radial;

}

double CColumn::GetHeight()

{

return height;

}

double CColumn::GetRadial()

{

return radial;

}

double CColumn::GetHeight()

{

return height;

}

DLL测试工程：

H文件：

#include "stdafx.h"

//导出函数

#pragma comment(lib,"G:\\其它\\VC驿站\\WinDLL\\Debug\\WinDLL.lib")

extern "C" float \_declspec(dllimport)calculate(float x,float y,int type);

//导出类

class \_declspec(dllimport) CColumn

{

public:

CColumn();

CColumn(double r,double h);

double GetArea();

double GetVolum();

double GetRadial();

double GetHeight();

void SetRadial(double r);

void SetHeight(double h);

protected:

private:

double radial;

double height;

};

CPP文件：

#include "stdafx.h"

#include <iostream>

#include "WinDLLTest.h"

using namespace std;

void test1();

void test2();

int \_tmain(int argc, \_TCHAR\* argv[])

{

int choice;

bool flag=true;

while (flag)

{

cout<<"请选择将要进行的实验：导出函数的调用（1），导出类的调用（2），退出(0):";

cin>>choice;

switch (choice)

{

case 1:

test1();

break;

case 2:

test2();

break;

default:

flag=false;

break;

}

}

return 0;

}

void test1()

{

int type;

float a,b;

while(1)

{

type=10;

cout<<"请选择计算类型：+(1),-(2),\*(3),/(4),退出(0)"<<endl;

while(type!=0&&type!=1&&type!=2&&type!=3&&type!=4)

cin>>type;

if (type==0)

break;

cout<<"请输入第一个数值：";

cin>>a;

cout<<"请输入第二个数值：";

cin>>b;

cout<<"运算结果："<<calculate(a,b,type)<<endl;

}

}

void test2()

{

int c=0;

bool flag=true;

CColumn co(0,0);

while (flag)

{

cout<<"请选择：创建一个圆柱体(1),退出(0):";

cin>>c;

switch(c)

{

case 1:

double r,h;

cout<<"请输入圆柱半径:r=";

cin>>r;

co.SetRadial(r);

cout<<"请输入圆柱高:h=";

cin>>h;

co.SetHeight(h);

cout<<"圆柱面积："<<co.GetArea()<<endl;

cout<<"圆柱体积："<<co.GetVolum()<<endl;

break;

default:

flag=false;

}

}

}