# 控制台窗口界面编程控制

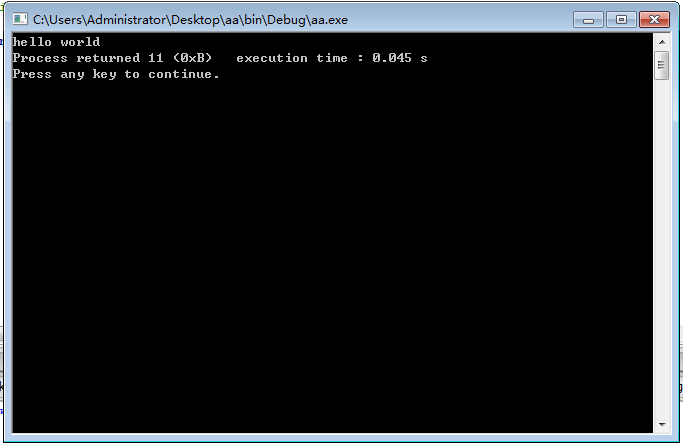
## (一) 概述操作

所谓控制台应用程序，就是指那些需要与传统DOS操作系统保持某种程序的兼容，同时又不需要为用户提供完善界面的程序。简单地讲，就是指Windows环境下运行的DOS程序。一旦控制台应用程序在Windows操作系统中运行后，就会弹出一个窗口。

例如下列代码：

|  |
| --- |
| #include <stdio.h>  int main()  {  printf("Hello, Console!\n");  return 0;  } |

单击小型编译工具栏中的“运行”按钮或按”F9”系统出现一个对话框，询问是否将此项目的工作文件夹设定源文件所在的文件夹，单击[是]按钮，系统开始编译。单击小型编译工具栏中的“Execute Program”按钮或按Ctrl+F键，运行刚才的程序。程序运行后，弹出下图的窗口: 如图所示，就是控制台窗口.



## (二) 控制台文本窗口的一般控制步骤

在Visual C++ 6.0中，控制台窗口界面的一般编程控制步骤如下：调用GetStdHandle获取当前的标准输入和标准输出

设备句柄。函数原型为：

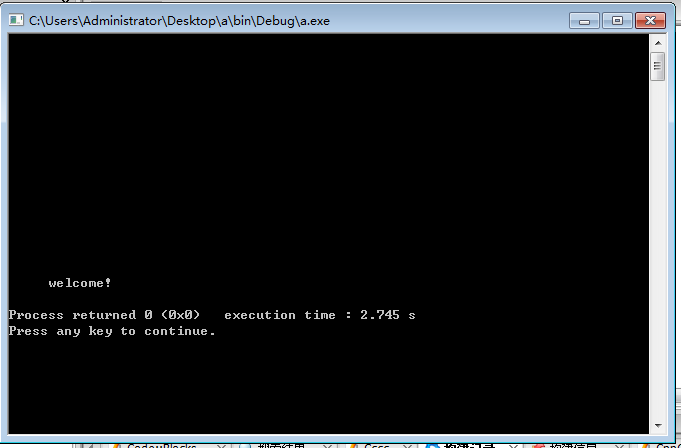
HANDLE GetStdHandle( DWORD nStdHandle );

其中，nStdHandle可以是STD\_INPUT\_HANDLE(标准输入设备句柄)、STD\_OUTPUT\_HANDLE(标准输出设备句柄)和STD\_ERROR\_HANDLE(标准错误句柄)。需要说明的是，“句柄”是Windows最常用的概念。它通常用来标识Windows资源(如菜单、图标、窗口等)和设备等对象。虽然可以把句柄理解为是一个指针变量类型，但它不是对象所在的地址指针，而是作为Windows系统内部表的索引值来使用的。调用相关文本界面控制的API函数。这些函数可分为三类。一是用于控制台窗口操作的函数(包括窗口的缓冲区大小、窗口前景字符和背景颜色、窗口标题、大小和位置等)；二是用于控制台输入输出的函数(包括字符属性操作函数)；其他的函数并为最后一类。调CloseHandle()来关闭输入输出句柄。注意，在程序中还必须包含头文件windows.h。

下面看一个程序：

|  |
| --- |
| /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \*如何在固定位置实现字符串welcome的打印  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  #include <stdio.h>  #include <stdlib.h>  #include <windows.h>//引入头文件  int main()  {  //定义句柄变量  HANDLE g\_ConsoleOutPut;  //定义COORD结构体变量pos并通过对其赋值设置光标坐标  COORD pos={5,15};  //获取一个句柄  g\_ConsoleOutPut=GetStdHandle(STD\_OUTPUT\_HANDLE);  //移动光标到pos指定位置  SetConsoleCursorPosition(g\_ConsoleOutPut,pos);  printf("welcome!\n");//输出字符  CloseHandle(g\_ConsoleOutPut); //关闭句柄  return 0;  } |

效果如图：



程序中，COORD是wincon.h定义的控制台结构体类型，其原型如下：

|  |
| --- |
| // 坐标结构体  typedef struct \_COORD {  SHORT X;  SHORT Y;  } COORD; |

## (三) 控制台窗口操作

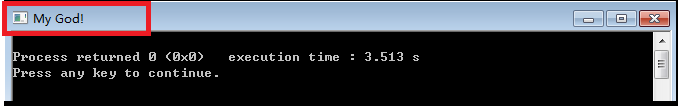
用于控制台窗口操作的API函数如下：

 SetConsoleTitle 设置控制台窗口标题

代码如下：

|  |
| --- |
| //如何设置控制台窗口标题  #include<stdio.h>  #include<stdlib.h>  #include <windows.h>  int main()  {  SetConsoleTitle("My God!");  return 0;  } |

效果如下：



需要说明的是，控制台窗口的原点坐标是(0, 0)，而最大的坐标是缓冲区大小减1，例如当缓冲区大小为80\*25时，其最大的坐标是(79, 24)。

## (四) 文本属性操作操作

与DOS字符相似，控制台窗口中的字符也有相应的属性。这些属性分为：文本的前景色、背景色和双字节字符集(DBCS)属性三种。事实上，我们最关心是文本颜色，这样可以构造出美观的界面。颜色属性都是一些预定义标识： FOREGROUND\_BLUE 蓝色 FOREGROUND\_GREEN 绿色 FOREGROUND\_RED 红色 FOREGROUND\_INTENSITY 加强 BACKGROUND\_BLUE 蓝色背景 BACKGROUND\_GREEN 绿色背景 BACKGROUND\_RED 红色背景 BACKGROUND\_INTENSITY 背景色加强 COMMON\_LVB\_REVERSE\_VIDEO 反色

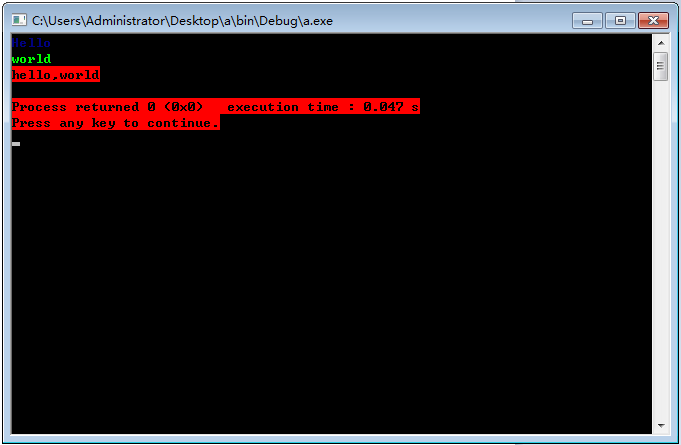
与文本属性相关的主要函数有：

|  |
| --- |
| HANDLE hConsoleOutput,  // 句柄  WORD wAttribute,  // 文本属性  COORD dwWriteCoord,  // 开始位置  BOOL SetConsoleTextAttribute// 设置WriteConsole等函数的字符属性  (HANDLE hConsoleOutput, /\*句柄\*/ WORD wAttributes //文本属性 ); |

## (五) 文本操作示例操作 下面看一个示例程序：

|  |
| --- |
| /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \*功能：在控制台上输出蓝色hello,绿色world  \*说明  \* 1）如何设置句柄变量  \* 2）如何通过GetStdHandle()获取一个句柄  \* 3）如何输出不同颜色字符串  \* 4)如何给hello，world增加红色背景  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  #include <stdio.h>  #include <stdlib.h>  #include <windows.h>//引入头文件  int main()  {  //设置句柄变量为g\_hConsoleOutput  HANDLE g\_ConsoleOutPut;  //获取一个句柄，参数设置为标准输出模式，并赋值给句柄变量  g\_ConsoleOutPut =GetStdHandle(STD\_OUTPUT\_HANDLE);  //设置输出字体颜色为蓝色  SetConsoleTextAttribute(g\_ConsoleOutPut,FOREGROUND\_BLUE);  printf("Hello\n");//设置  //设置字体为高亮绿色  SetConsoleTextAttribute(g\_ConsoleOutPut,\  FOREGROUND\_INTENSITY|FOREGROUND\_GREEN);  printf("world\n");//输出  //”设置字体hello,world”为高亮并添加红色背景  SetConsoleTextAttribute(g\_ConsoleOutPut,\  BACKGROUND\_INTENSITY| BACKGROUND\_RED);  printf(“hello,world\n”);  CloseHandle(g\_ConsoleOutPut);  return 0;  } |

效果如下：



背景颜色:

0x10:深蓝

0x20:深绿

0x30:灰蓝

0x40:暗红

0x50:紫色

0x60:暗黄

0x70:灰白

0x80:灰色

0x90:蓝色

0xa0:嫩绿

0xb0:薄荷绿

0xc0:红色

0xd0:粉色

0xe0:黄色

0xf0:白色