Console Output

It appears you're looking for an explanation of the WriteConsole function and some associated data structures. Below, I'll provide explanations for WriteConsole and the COORD and SMALL_RECT structures.

WriteConsole Function:

WriteConsole is a Win32 function used to write a string to the console window at the current cursor position. It is used for console output. Here's a breakdown of its parameters:

- **hConsoleOutput:** This is the handle to the console output stream.
- **lpBuffer:** It's a pointer to the array of characters you want to write. nNumberOfCharsToWrite: This parameter holds the length of the array you want to write.
- **lpNumberOfCharsWritten:** It's a pointer to an integer that will receive the number of characters written when the function returns.
- **lpReserved:** This parameter is not used, so you can set it to zero.

WriteConsole writes the string and advances the cursor just past the last character written. It can handle standard ASCII control characters like tabs, carriage returns, and line feeds, and the string doesn't need to be null-terminated.

COORD Structure:

The COORD structure is used in various Win32 console functions. It represents the coordinates of a character cell in the console screen buffer.

The origin of this coordinate system is at the top left cell of the console screen. The structure has two fields:

- **X:** This field is of type WORD and represents the X-coordinate.
- **Y:** This field is also of type WORD and represents the Y-coordinate.

SMALL_RECT Structure:

The SMALL_RECT structure is another data structure used in Win32 console functions. It specifies a rectangular region by defining the upper left and lower right corners of the rectangle within the

console window.

It's useful for specifying character cells in the console window. The structure has the following fields:

- **Left:** This field is of type WORD and represents the left coordinate of the rectangle.
- **Top:** This field, also of type WORD, represents the top coordinate.
- **Right:** This field, again of type WORD, represents the right coordinate.
- **Bottom:** This field, once more of type WORD, represents the bottom coordinate.

These data structures are used in various console-related Win32 functions to manage and manipulate console windows. The WriteConsole function is particularly useful for writing content to the console.

Example 1: WriteConsole function

```
; Win32 Console Example #1(Console1.asm)
; This program calls the following Win32 Console functions:
; GetStdHandle, ExitProcess, WriteConsole
INCLUDE Irvine32.inc
.data
endl EQU <0dh,0ah>       ; End of line sequence
message LABEL BYTE
    BYTE "This program is a simple demonstration of"
    BYTE "console mode output, using the GetStdHandle"
    BYTE "and WriteConsole functions.",endl
messageSize DWORD ($ - message)
consoleHandle HANDLE 0    ; Handle to standard output device
bytesWritten DWORD ?      ; Number of bytes written
.code
main PROC
    ; Get the console output handle:
    INVOKE GetStdHandle, STD_OUTPUT_HANDLE
    mov consoleHandle,eax

    ; Write a string to the console:
    INVOKE WriteConsole,
        consoleHandle,   ; Console output handle
        ADDR message,    ; String pointer
        messageSize,     ; String length
        ADDR bytesWritten, ; Returns number of bytes written
        0                ; Not used

    ; Exit the program:
    INVOKE ExitProcess, 0
main ENDP
END main
```

The provided program, Console1.asm, is a simple example that demonstrates the use of the GetStdHandle, WriteConsole, and ExitProcess functions in a Win32 console application. Here's a breakdown of the code:

The .data section begins by defining an endl constant that represents the end-of-line sequence (carriage return and line feed).

A message is defined using the message label. It contains a multiline text string that the program will write to the console. The messageSize variable is used to store the size of the message string.

The consoleHandle variable is declared as a HANDLE and initialized to 0. This variable will hold the handle to the standard output device (console).

bytesWritten is declared as a DWORD and will be used to store the number of bytes written by the WriteConsole function.

The .code section contains the main procedure.

INVOKE GetStdHandle, STD_OUTPUT_HANDLE is used to obtain the handle to the standard output (the console). The obtained handle is stored in the consoleHandle variable.

The **INVOKE WriteConsole function** is called to write the message string to the console. It takes the following parameters:

- **consoleHandle:** The handle to the console output.
- **ADDR message:** A pointer to the message string.
- **messageSize:** The length of the message string.
- **ADDR bytesWritten:** A pointer to a variable that will receive the number of bytes written.
- **0:** An unused parameter.
- Finally, **INVOKE ExitProcess, 0** is called to exit the program.

When you run this program, it will write the message to the console, and the console window will display the content of the message string.

The output will look like this:

```
This program is a simple demonstration of
console mode output, using the GetStdHandle
and WriteConsole functions.
```

This code demonstrates how to use the **Win32 Console functions** to write a message to the console window. The message is stored in the message variable and is written to the console using the **WriteConsole function**. Finally, the program exits using ExitProcess

WriteConsoleOutputCharacter function

```
311 INCLUDE Irvine32.inc
312
313 .data
314     message BYTE "Hello, World!",0  ; The string to be written
315     coord COORD <5, 5>              ; Starting coordinates in the console
316
317 .code
318 main PROC
319     ; Get the console output handle:
320     INVOKE GetStdHandle, STD_OUTPUT_HANDLE
321     mov edi, eax  ; Store the console output handle in EDI
322
323     ; Write the message to the console at the specified coordinates:
324     INVOKE WriteConsoleOutputCharacter, edi, ADDR message, LENGTHOF message - 1, coord, NULL
325
326     ; Exit the program
327     INVOKE ExitProcess, 0
328
329 main ENDP
330
331 END main
```

The WriteConsoleOutputCharacter function in Win32 allows you to copy an array of characters to consecutive cells in the console screen buffer at a specified location. Here's a breakdown of its parameters:

- **hConsoleOutput:** This is the handle to the console output. It specifies the console screen buffer where you want to write the characters.
- **lpCharacter:** A pointer to the buffer containing the characters you want to write.
- **nLength:** The size of the buffer, indicating the number of characters to write from the buffer.
- **dwWriteCoord:** This parameter specifies the coordinates of the first cell where you want to start writing. It is of type COORD, which holds X (column) and Y (row) coordinates.
- **lpNumberOfCharsWritten:** A pointer to a DWORD that will receive the count of characters written by the function.

This function writes the characters to the console screen buffer, and if the text reaches the end of a line, it wraps around to the next line.

It's important to note that this function doesn't change the attribute values in the screen buffer, and it ignores ASCII control codes such as tab, carriage return, and line feed.

If the function is successful, it returns a non-zero value, and the number of characters written is stored in lpNumberOfCharsWritten. If it fails, it returns zero.

You can use WriteConsoleOutputCharacter to write text directly to the console, which can be useful for more advanced console applications.