Short test – basic knowledge C++ and SQL

**Prerequisits:**

* C++ compiler / linker
* Standard c libraries / boost / STL / MFC / VCL
* No preference for any framework / clean and “easy” code is the goal
* NO INTERNET CONNECTION

**#1 SQL:**

* Describe the difference between “LEFT JOIN” and “INNER JOIN”

**An INNER JOIN retrieves only the rows that have matching values in both tables being joined based on the condition provided.**

**A Left join selects all the rows from the left table and matching rows from the right table based on the condition provided.**

* Which of both “can” result in a larger result set, if the rest of the query is identical and why?

**The left join results in a much larger dataset compared to inner join as it retrieves all the records from the left table but inner join retrieves only the records which matches the condition provided.**

**#2 SQL:**

* Given tables: (id is always the primary key of the table, strings are indicated by quotation marks)

Tablename: customerdata

Columns: id,tid,name // (tid is just a special device identifier)

Row 1: 1,”69500000”,”Martin”

Row 2: 2,”69500001”,”Christian”

Tablename: card

Columns: id,name

Row 1: 1,”MASTER”

Row 2: 3,”VISA”

Tablename: customercards

Columns: id,customerdata\_id,card\_id,contractnumber (card\_id refers to card.id, customerdata\_id refers to customerdata.id)

Row 1: 1,1,1,”ABC123”

Row 2: 2,1,3,”DEF456”

Row 3: 3,2,1,”GHI789”

* **Create a SQL query, that will produce THIS output:**

Columns: tid, name of customerdata, name of VISA card, visa contractnumber, name of Master card, mastercard contractnumber

Row 1: “69500000”,”Martin”,”VISA”,”DEF456”,”MASTER”,”ABC123”

Row 2: “69500001”,”Christian”,”VISA”,<null>,”MASTER”,”GHI789”

Comment: You can use constants ‘1’ and ‘3’ for VISA and MASTER – no other constants are allowed

Comment: please sort by TID

**SELECT cd.tid,cd.name AS "Name of Customerdata",c1.name AS "Name of VISA card",cc1.contractnumber AS "visa contractnumber",c2.name AS "name of Master card",cc2.contractnumber AS "mastercard contractnumber" FROM customerdata AS cd**

**LEFT JOIN customercards AS cc1 ON cd.id = cc1.customerdata\_id LEFT JOIN card AS c1 ON cc1.card\_id = c1.id AND c1.name = 'VISA LEFT JOIN customercards AS cc2 ON cd.id = cc2.customerdata\_id LEFT JOIN card AS c2 ON cc2.card\_id = c2.id AND c2.name = 'MASTER';**

* **Advanced: Can you identify a second possibility to create this output? If yes, just describe it with a few sentences – no query needed**

**I usually use JOINS in this kind of situation as it is less complex and easily understood by other developers moreover I dont have SQL SERVER installed in my computer so can’t comment on this**

**#3 SQL:**

* Use the same tables as in #2
* Please create with one query a resultset like this:

Columns: tid, number of card contracts for this customer (how many card configurations are present for this customer in in table customercards?)

Row 1: “69500000”, 2

Row 2: “69500001”, 1

**I believe this can be achieved by aggregate functions like GROUP BY and COUNT**

Comment: Please sort the result in the way, that the customers with the highest number of card contracts is on top

**#4 C++:**

* What is the purpose of ‘delete’ operator?
* Which members of the base class are visible in a derived class?
* **The delete operators deallocate memory dynamically allocated using new operator**
* **members declared under protected and public access modifiers are derived to child class, but protected members cannot be accessible via object.**

**#5 C++:**

* Please create a little subroutine, that will convert an integer (32 bit unsigned) into a “BCD” format with variable length (if it does not fit, your subroutine should abort the conversion)
* BCD background:

In a “binary compressed data” string, every number (0 – F) is stored in a “nibble” (4 Bit). One byte (8 bits = unsigned char) can carry 2 numbers. Example: integer value 98 would fit into one byte: unsigned char b=0x98; (or decimal = 152).

! BCD strings can only contain numbers 0-9 and the characters A-F – otherwise it would not fit into a nibble.

Function header:

unsigned int convert\_int\_to\_bcd(unsigned char\* destination\_buffer,usigned int bufferlength, unsigned int value);

Parameters:

[out] unsigned int = used buffer size in bytes (0 in case of error or equal to bufferlength, if successful)

[in] destination\_buffer = please write the result into this buffer

[in] bufferlength = length of buffer – please do not convert the value, if this buffer is too small

The length of the buffer must be used completely – it indicates how many bytes should be used for the new BCD string (prefill with zeros)

[in] value = value to convert into BCD format

Example value: value = 9873432, bufferlength = 4 (bytes)

Example result: 4 (HEXDUMP): 09873432 // destination\_buffer[0]==0x09; destination\_buffer[1]==0x87; …

**#6 C++:**

* Create a little subroutine that sorts an integer vector in descending order.
* Input parameters: pointer to an array, number of elements
* Result: no result, just sort the array.

e.g. int a[5]; (content:{5,-3,1,9,2} after sort: {9,5,2,1,-3})

**#include <iostream>**

**void dec\_sort(int\* arr, int total\_elements) {**

**for (int i = 0; i < total\_elements - 1; i++) {**

**for (int j = 0; j < total\_elements - i - 1; j++) {**

**if (arr[j] < arr[j + 1]) {**

**int temp = arr[j];**

**arr[j] = arr[j + 1];**

**arr[j + 1] = temp;**

**}**

**}**

**}**

**}**

**int main() {**

**int arr[] = {5,-3,1,9,2};**

**int total\_elements = sizeof(arr) / sizeof(arr[0]);**

**dec\_sort(arr, total\_elements);**

**for (int i = 0; i < total\_elements; i++) {**

**std::cout << arr[i] << " ";**

**}**

**return 0;**

**}**

**// Same output can achieved using STL sort**

#7 C++:

* Create a subroutine, that prints out (console? GUI? File – does not matter) a complete directory tree (displaying also the files and directories of subfolders)
* Please use standard Windows routines (FindFirstFile, FindNextFile, FindClose – see next page) – include <windows.h>
* INPUT PARAMATERS: a string, indicating the PATH to start with
* OUTPUT VALUE: an integer, indicating the number of printed lines
* No SORT needed
* Example: Start path: “c:\\windows\\debug\\”

Result (line count = 6):

Line 1: Setup

Line 2: Setup\Backup

Line 3: Setup\Backup\INTPPM\_Backup

Line 4: Setup\UpdSh.log

Line 5: passwd.log

Line 6: NetSetup.log

DONE!

**#include "pch.h"**

**#include "framework.h"**

**#include "Project1.h"**

**#ifdef \_DEBUG**

**#define new DEBUG\_NEW**

**#endif**

**// The one and only application object**

**CWinApp theApp;**

**using namespace std;**

**int printDirectoryTree(LPCWSTR path, const std::wstring& currentPath) {**

**WIN32\_FIND\_DATAW findFileData;**

**HANDLE hFind = FindFirstFileW((std::wstring(path) + L"\\\*").c\_str(), &findFileData);**

**if (hFind == INVALID\_HANDLE\_VALUE) {**

**std::wcerr << L"Error: Invalid directory path." << std::endl;**

**return 0;**

**}**

**static int lineCount = 0;**

**do {**

**if (wcscmp(findFileData.cFileName, L".") != 0 && wcscmp(findFileData.cFileName, L"..") != 0) {**

**//if directories have to be listed first then files and directories to be moved into a seperate arrays**

**//and print its content after function is executed**

**std::wcout << L"Line " << ++lineCount << L": " << currentPath << L"\\" << findFileData.cFileName << std::endl;**

**if (findFileData.dwFileAttributes & FILE\_ATTRIBUTE\_DIRECTORY) {**

**std::wstring subDirPath = std::wstring(path) + L"\\" + findFileData.cFileName;**

**printDirectoryTree(subDirPath.c\_str(), currentPath + L"\\" + findFileData.cFileName);**

**}**

**}**

**} while (FindNextFileW(hFind, &findFileData) != 0);**

**FindClose(hFind);**

**return lineCount;**

**}**

**int main()**

**{**

**int nRetCode = 0;**

**HMODULE hModule = ::GetModuleHandle(nullptr);**

**if (hModule != nullptr)**

**{**

**// initialize MFC and print and error on failure**

**if (!AfxWinInit(hModule, nullptr, ::GetCommandLine(), 0))**

**{**

**// TODO: code your application's behavior here.**

**wprintf(L"Fatal Error: MFC initialization failed\n");**

**nRetCode = 1;**

**}**

**else**

**{**

**LPCWSTR startPath = L"C:\\tmp\\John";**

**int lineCount = printDirectoryTree(startPath, startPath);**

**std::wcout << L"DONE!" << std::endl;**

**std::wcout << L"Total line count = " << lineCount << std::endl;**

**}**

**}**

**else**

**{**

**// TODO: change error code to suit your needs**

**wprintf(L"Fatal Error: GetModuleHandle failed\n");**

**nRetCode = 1;**

**}**

**return nRetCode;**

**}**

# FindFirstFile Function

Searches a directory for a file or subdirectory with a name that matches a specific name (or partial name if wildcards are used).

### Syntax

HANDLE WINAPI FindFirstFile(

\_\_in   LPCTSTR lpFileName,

\_\_out  LPWIN32\_FIND\_DATA lpFindFileData

);

### Parameters

*lpFileName* [in]

The directory or path, and the file name, which can include wildcard characters, for example, an asterisk (\*) or a question mark (?).

This parameter should not be NULL, an invalid string (for example, an empty string or a string that is missing the terminating null character), or end in a trailing backslash (\).

If the string ends with a wildcard, period (.), or directory name, the user must have access permissions to the root and all subdirectories on the path.

*lpFindFileData* [out]

A pointer to the [**WIN32\_FIND\_DATA**](http://msdn.microsoft.com/en-us/library/aa365740%28v=vs.85%29.aspx) structure that receives information about a found file or directory.

### Return Value

If the function succeeds, the return value is a search handle used in a subsequent call to [**FindNextFile**](http://msdn.microsoft.com/en-us/library/aa364428%28v=vs.85%29.aspx) or [**FindClose**](http://msdn.microsoft.com/en-us/library/aa364413%28v=vs.85%29.aspx), and the *lpFindFileData* parameter contains information about the first file or directory found.

If the function fails or fails to locate files from the search string in the *lpFileName* parameter, the return value is INVALID\_HANDLE\_VALUE and the contents of *lpFindFileData* are indeterminate. To get extended error information, call the [**GetLastError**](http://msdn.microsoft.com/en-us/library/ms679360%28v=vs.85%29.aspx) function.

If the function fails because no matching files can be found, the [**GetLastError**](http://msdn.microsoft.com/en-us/library/ms679360%28v=vs.85%29.aspx) function returns ERROR\_FILE\_NOT\_FOUND.

# APPENDIX…

# WIN32\_FIND\_DATA Structure

Contains information about the file that is found by the [**FindFirstFile**](http://msdn.microsoft.com/en-us/library/aa364418%28v=vs.85%29.aspx), [**FindFirstFileEx**](http://msdn.microsoft.com/en-us/library/aa364419%28v=vs.85%29.aspx), or [**FindNextFile**](http://msdn.microsoft.com/en-us/library/aa364428%28v=vs.85%29.aspx) function.

### Syntax

typedef struct \_WIN32\_FIND\_DATA {

DWORD    dwFileAttributes;

FILETIME ftCreationTime;

FILETIME ftLastAccessTime;

FILETIME ftLastWriteTime;

DWORD    nFileSizeHigh;

DWORD    nFileSizeLow;

DWORD    dwReserved0;

DWORD    dwReserved1;

TCHAR    cFileName[MAX\_PATH];

TCHAR    cAlternateFileName[14];

} WIN32\_FIND\_DATA, \*PWIN32\_FIND\_DATA, \*LPWIN32\_FIND\_DATA;

### Members

**dwFileAttributes**

The file attributes of a file.

For possible values and their descriptions, see [File Attribute Constants](http://msdn.microsoft.com/en-us/library/gg258117%28v=vs.85%29.aspx).

**ftCreationTime**

A [**FILETIME**](http://msdn.microsoft.com/en-us/library/ms724284%28v=vs.85%29.aspx) structure that specifies when a file or directory was created.

If the underlying file system does not support creation time, this member is zero.

**ftLastAccessTime**

A [**FILETIME**](http://msdn.microsoft.com/en-us/library/ms724284%28v=vs.85%29.aspx) structure.

For a file, the structure specifies when the file was last read from, written to, or for executable files, run.

For a directory, the structure specifies when the directory is created. If the underlying file system does not support last access time, this member is zero.

On the FAT file system, the specified date for both files and directories is correct, but the time of day is always set to midnight.

**ftLastWriteTime**

A [**FILETIME**](http://msdn.microsoft.com/en-us/library/ms724284%28v=vs.85%29.aspx) structure.

For a file, the structure specifies when the file was last written to, truncated, or overwritten, for example, when [**WriteFile**](http://msdn.microsoft.com/en-us/library/aa365747%28v=vs.85%29.aspx) or [**SetEndOfFile**](http://msdn.microsoft.com/en-us/library/aa365531%28v=vs.85%29.aspx) are used. The date and time are not updated when file attributes or security descriptors are changed.

For a directory, the structure specifies when the directory is created. If the underlying file system does not support last write time, this member is zero.

**nFileSizeHigh**

The high-order **DWORD** value of the file size, in bytes.

This value is zero unless the file size is greater than MAXDWORD.

The size of the file is equal to (**nFileSizeHigh** \* (MAXDWORD+1)) + **nFileSizeLow**.

**nFileSizeLow**

The low-order **DWORD** value of the file size, in bytes.

**dwReserved0**

If the **dwFileAttributes** member includes the FILE\_ATTRIBUTE\_REPARSE\_POINT attribute, this member specifies the reparse point tag.

Otherwise, this value is undefined and should not be used.

For more information see [Reparse Point Tags](http://msdn.microsoft.com/en-us/library/aa365511%28v=vs.85%29.aspx).

IO\_REPARSE\_TAG\_DFS (0x8000000A)

IO\_REPARSE\_TAG\_DFSR (0x80000012)

IO\_REPARSE\_TAG\_HSM (0xC0000004)

IO\_REPARSE\_TAG\_HSM2 (0x80000006)

IO\_REPARSE\_TAG\_MOUNT\_POINT (0xA0000003)

IO\_REPARSE\_TAG\_SIS (0x80000007)

IO\_REPARSE\_TAG\_SYMLINK (0xA000000C)

**dwReserved1**

Reserved for future use.

**cFileName**

The name of the file.

**cAlternateFileName**

An alternative name for the file.

This name is in the classic 8.3 file name format.

# FindNextFile Function

Continues a file search from a previous call to the [**FindFirstFile**](http://msdn.microsoft.com/en-us/library/aa364418%28v=vs.85%29.aspx) function.

### Syntax

BOOL WINAPI FindNextFile(

\_\_in   HANDLE hFindFile,

\_\_out  LPWIN32\_FIND\_DATA lpFindFileData

);

### Parameters

*hFindFile* [in]

The search handle returned by a previous call to the [**FindFirstFile**](http://msdn.microsoft.com/en-us/library/aa364418%28v=vs.85%29.aspx) function.

*lpFindFileData* [out]

A pointer to the [**WIN32\_FIND\_DATA**](http://msdn.microsoft.com/en-us/library/aa365740%28v=vs.85%29.aspx) structure that receives information about the found file or subdirectory.

The structure can be used in subsequent calls to **FindNextFile** to indicate from which file to continue the search.

### Return Value

If the function succeeds, the return value is nonzero and the *lpFindFileData* parameter contains information about the next file or directory found.

If the function fails, the return value is zero and the contents of *lpFindFileData* are indeterminate. To get extended error information, call the [**GetLastError**](http://msdn.microsoft.com/en-us/library/ms679360%28v=vs.85%29.aspx) function.

If the function fails because no more matching files can be found, the [**GetLastError**](http://msdn.microsoft.com/en-us/library/ms679360%28v=vs.85%29.aspx) function returns ERROR\_NO\_MORE\_FILES.

### Remarks

This function uses the same search filters that were used to create the search handle passed in the *hFindFile* parameter. For additional information, see [**FindFirstFile**](http://msdn.microsoft.com/en-us/library/aa364418%28v=vs.85%29.aspx) and [**FindFirstFileEx**](http://msdn.microsoft.com/en-us/library/aa364419%28v=vs.85%29.aspx).

The order in which the search returns the files, such as alphabetical order, is not guaranteed, and is dependent on the file system. If the data must be sorted, the application must do the ordering after obtaining all the results.

# FindClose Function

Closes a file search handle opened by the [**FindFirstFile**](http://msdn.microsoft.com/en-us/library/aa364418%28v=vs.85%29.aspx), [**FindFirstFileEx**](http://msdn.microsoft.com/en-us/library/aa364419%28v=vs.85%29.aspx), [**FindFirstFileNameW**](http://msdn.microsoft.com/en-us/library/aa364421%28v=vs.85%29.aspx), [**FindFirstFileNameTransactedW**](http://msdn.microsoft.com/en-us/library/aa364420%28v=vs.85%29.aspx), [**FindFirstFileTransacted**](http://msdn.microsoft.com/en-us/library/aa364422%28v=vs.85%29.aspx), [**FindFirstStreamTransactedW**](http://msdn.microsoft.com/en-us/library/aa364423%28v=vs.85%29.aspx), or [**FindFirstStreamW**](http://msdn.microsoft.com/en-us/library/aa364424%28v=vs.85%29.aspx) functions.

### Syntax

BOOL WINAPI FindClose(

\_\_inout  HANDLE hFindFile

);

### Parameters

*hFindFile* [in, out]

The file search handle.

### Return Value

If the function succeeds, the return value is nonzero.

If the function fails, the return value is zero. To get extended error information, call [**GetLastError**](http://msdn.microsoft.com/en-us/library/ms679360%28v=vs.85%29.aspx).