C++DBLIB

The library, which combines C++ and the MS/SYBASE SQLSERVER[1] in an ideal manner.

- **Elegant and simple** access on tables and columns of the SQLSERVER´s databases
- **Each SQL data type** has a corresponding predefined class i.e.: smallint↔CDbColSmall, image↔CDbColImage, etc. That means compatible date/time and money arithmetic!
- **Stream concept** for reading and writing data from and to a SQLSERVER database when using this (predefined) classe.
- **Strict type checking** when sending SQL statements to the SERVER using the overloaded I/O-stream class.
- **Corresponding classes:** The columns of a table or view are mapped on a class using our tool DBGEN.EXE. Each data member of the class has a corresponding column in the respective table or view.
- **Default member functions**: Each class you create with DBGEN includes a set of standard member functions for row retrieving and manipulating.
- **Record interface:** You can read ,write and update a single row as a whole - if the table has a unique key - with a single function (f.example : .Custom.Select(5678); ...)
- **Record list interface:** If the result of your query consists of more than one row, the incoming rows will be stored successivly (if you use the recordlist class).
- **Storde procedure mapping**: Each stored procedure of your database can be mapped to a member function of a dedicated class (=table). I.e.: a member function will be created with the same parameters the Stored Procedure has. (Strict type checking of parameters!).
- **Exeception handling** (special Execption class with TRY, CATCH and THROW).
- **Transaction handling.**
- **High performance!!**

You need MS-Visual C++ 10, (MFC 2.0), SYBASE or MS/SQLSERVER[2], the DB-Library for MS-DOS or MS-WINDOWS 3.1 Version 4.20.21 or greater.

If you want more detailed informations write, call or send a message to:

Klem, Mülller & Heinrich Gbr, Ottenbrucher Str. 53 42105 Wuppertal, **GERMANY**, Tel/FAX.: +49/0202/31 87 30

[1] MS-SQLSERVER, MS-DOS, MS-WINDOWS ar trademarks of the Microsoft corporation.

[2] see footmark one

A simple example shows the elegance of our implementation:
(much more examples with record lists usw when you call for papers)

```
#include <assert.h>
#include "cdblib.h"

int main ()
{
        if (!CDbConnection::InitSystem()) {
                cerr << "\nGeneral network error\n";
        return 100;

        CDbConnection Con(1);   // Number of required db-processes (=1)
        CDbChannel C;           // create a channel object for writing and
        // reading data
        CDbColStr S1;           // create an object of type (CDbCol)String
                                        // this is the equivalent of char or varchar
        CDbColTime S2;                  // create a date/time object
        TRY {                                   // if connection failes goto catch block
                Con.Connect("server",   // initialize connection object with
                                "SA",           // server name, user name and database
                                "secret");      // and password
                C.Connect(Con);         // now try to connect ...
                C << "select GetDate(),GetDate()"; // send statement to server
                                        //- no execution up to now
                C.Go(1);                                        // carry out 1st statement
                                                                // in command buffer
        C >> S1 >> S2;                          // get results and put them
                                        // corresponding objects
                cout  << "Result : "                    // show data on screen
                        << endl
                        << "CDbColStr ("
                        << S1 << ")"
                        << endl;
                cout << "CDbColTime (" << S2 << ")" << endl;
        }
        CATCH (CDbEx,e) {
                cerr << "\ndatabase error\n";
        return 1;
        }
        END_CATCH
        return 0;
```