

## Access till PostgreSQL från C/C++

Alla större databashanterare erbjuder ett interface för att administrera databasen och ställa databasfrågor från högnivåspråk som Java, C/C++, Python m.fl. I Java t.ex. används JDBC (Java Database Connectivity) som ger ett standardiserat interface till olika databaser. I den här övningen ska vi använda API:et `pqlib` från PostgreSQL för att accessa en databas inifrån ett C++-program. I korthet består `pqlib` av en header fil, `libpq-fe.h`, som definierar C-funktioner, konstanter mm samt ett antal lib-filer som innehåller de kompilerade funktionerna. Headerfilen ska inkluderas vid kompilering och lib-filerna ska tillföras vid länkeningen av programmet. I den här övningen används ett mycket begränsat antal funktioner för att kunna ansluta till databasen, ställa sql-frågor och hämta resultaten:

```
PGconn* PQconnectdb(const char *conninfo);
```

Skapar en förbindelse till databasen, `conninfo` är en sträng med inloggningsinformation. Resultatet returneras som en pekare till `PGconn` struct som sedan skickas med som argument till andra PQ-funktioner.

Typiskt för `pqlib` är att resultatet ofta fås som en pekare till en struct som funktionen har allokerat dynamiskt. När den inte längre behövs åligger det klientkoden att deallokera minne och det görs genom anrop till `PQclear` (pekare till struct).

```
ConnStatusType PQstatus(PGconn *)
```

 kontrollerar status för en förbindelse och returnerar resultatet i en `ConnStatusType` som kan vara `CONNECTION_OK` eller `CONNECTION_BAD`. Vid feltilstånd, t.ex. efter `CONNECTION_BAD` kan ett felmeddelande fås genom `PQerrorMessage` (`PGconn *`) som returnerar en C-sträng.

SQL-frågor görs genom

```
PGresult * PQexec(PGconn *conn, const char *sqlstring);
```

där returvärdet antingen är en pekare till resultatet eller en `nullptr`.

En SQL-fråga mot en databas som har flera samtidiga användare görs som en transaktion för att resultatet inte ska kunna påverkas av andra parallella accesser till samma tabeller. Före varje fråga gör vi därför ett anrop av typen

```
PGresult *res = PQexec(dbConnection, "BEGIN");
```

 och när vi är färdiga med resultatet anropar vi

```
PGresult *res = PQexec(dbConnection, "END");
```

 Dessa båda anrop ramar in ett transaktionsblock där vi gör våra SQL-frågor.

Om `PGresult`-pekaren inte är en `nullptr` så kan vi via pekaren hämta svaren.

```
PQntuples(PGresult *)
```

 returnerar antal rader i svaret och

```
PQnfields(PGresult *)
```

 returnerar antalet kolumner.

```
PQfname(PGresult *, i)
```

 returnerar namnet på den *i*:te kolumnen som en C-string och

```
PQgetvalue(PGresult *, i, j)
```

 returnerar värdet på *i*:te raden *j*:te kolumnen som en C-string.

Fullständig dokumentation för `pqlib` finns på

<http://www.postgresql.org/docs/9.5/interactive/libpq.html>

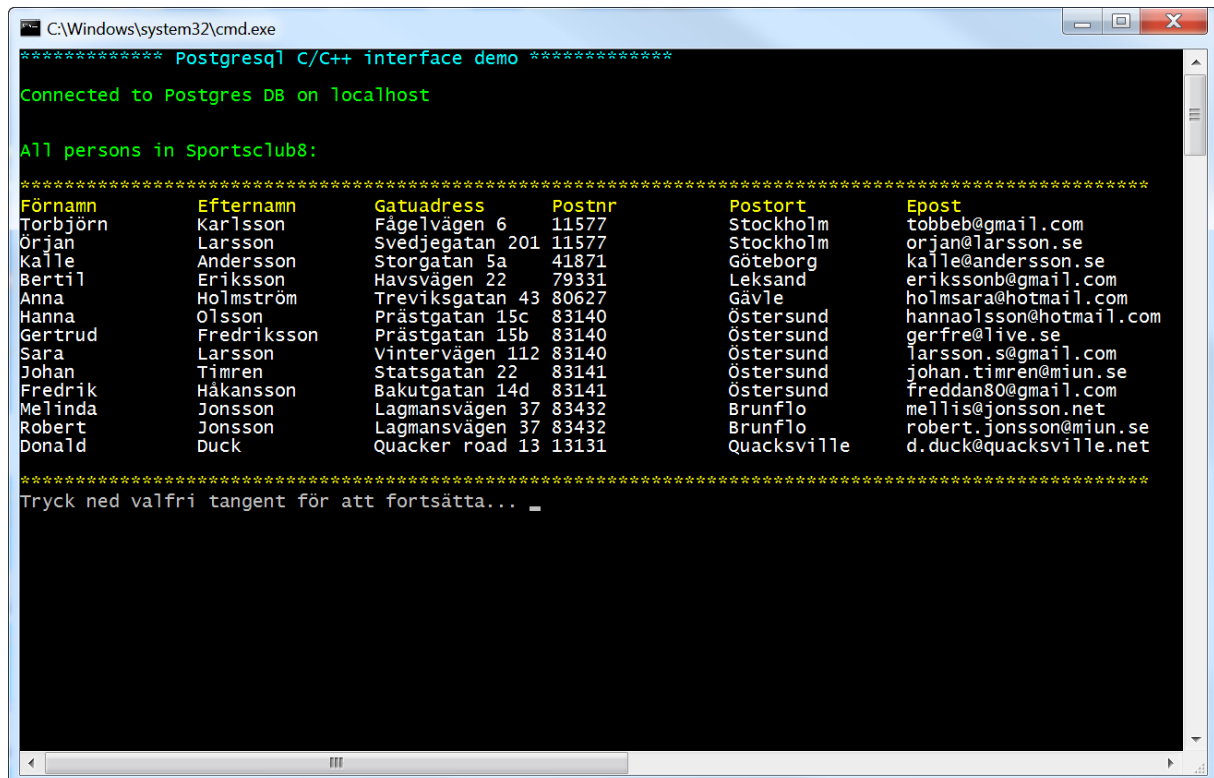
I den medföljande koden hittar du klassen `DBhandler` som kapslar in dessa grundläggande API-funktioner i några medlemsfunktioner för att underlätta användandet. Dokumentation finns i `DBhandler.h`.

Innan PQLib kan användas i ett C++-program måste en del förberedelser, konfiguration och anpassning av projektet göras. Detta finns beskrivet för Windows (VisualStudio) i den medföljande filen `VS-pqlib-config.pdf`. Det medföljande VisualStudio-projektet fungerar "out of the box" om du har 32-bitarsversionen av PostgreSQL installerad.

## Uppgift

Uppgiften bygger på databasen som finns i backup-filen `sportsclub8.sql` som du ska ladda in i din lokala databasserver. Öppna VisualStudio-projektet `DBtest.vcxproj`. I main-filen `DBtest.cpp` sätter du `PG_PASSWORD` i

`const string PG_PASSWORD { "XXXXXXX" };` till ditt eget lösenord för postgres. När du sedan kompilerar och kör programmet blir resultatet något som liknar det här:



```
C:\Windows\system32\cmd.exe
***** PostgreSQL C/C++ interface demo *****

Connected to Postgres DB on localhost

All persons in sportsclub8:

*****
Förnamn      Efternamn    Gatadress     Postnr       Postort      Epost
Torbjörn     Karlsson     Fågelvägen 6  11577        Stockholm    tobbeg@gmail.com
Örjan        Larsson      Svedjegatan 201 11577        Stockholm    orjan@larsson.se
Kalle        Andersson    Storgatan 5a   41871        Göteborg     kalle@andersson.se
Bertil       Eriksson     Havsvägen 22   79331        Leksand       erikssonb@gmail.com
Anna         Holmström    Treviksgatan 43 80627        Gävle         holmsara@hotmail.com
Hanna        Olsson       Prästgatan 15c 83140        Östersund     hannaolsson@hotmail.com
Gertrud      Fredriksson  Prästgatan 15b 83140        Östersund     gerfre@live.se
Sara         Larsson      Vintervägen 112 83140        Östersund     larsson.s@gmail.com
Johan        Timren       Statsgatan 22  83141        Östersund     johan.timren@miun.se
Fredrik      Håkansson    Bakutgatan 14d 83141        Östersund     freddan80@gmail.com
Melinda      Jonsson      Lagmansvägen 37 83432        Brunflo       mellis@jonsson.net
Robert       Jonsson      Lagmansvägen 37 83432        Brunflo       robert.jonsson@miun.se
Donald       Duck         Quacker road 13 13131        Quacksville   d.duck@quacksville.net
*****

Tryck ned valfri tangent för att fortsätta... _
```

Du behöver förmodligen ändra egenskaperna för console-fönstret. För att ändra storleken på fönstret högerklickar du på fönstrets övre ram och väljer Egenskaper. Sätt Tecken till Lucida console och under Layout sätter du fönstrets bredd till 105 tecken och höjden till 45 rader. Stäng och kör programmet igen så ska resultatet bli ungefär som bilden här ovan.

Som du ser så tillhör Donald Duck denna klubb och din uppgift är att utveckla programmet så att endast Donald kan se listan på medlemmar. För att kunna göra det måste han först ange sitt förnamn och sin epostadress, sedan ska han ange sitt lösenord som är **kalle**. I databasen

ligger lösenorden krypterade med hash-algoritmen **md5**. För att kontrollera att rätt lösenord är inmatat måste du alltså först kryptera det med md5-algoritmen. Du kan använda den inbyggda funktionen md5 som finns i PostgreSQL för att göra detta. Funktionen tar den okrypterade strängen som argument och returnerar den krypterade strängen.

Ett inloggningsförsök som misslyckas beroende på att en eller flera av uppgifterna är felaktig ske ge följande resultat:

```
C:\Windows\system32\cmd.exe
***** PostgreSQL C/C++ interface demo *****

Connected to Postgres DB on localhost

Enter first name and email address for the duck!
Donald kalle.duck@kvacksvill.net
Enter password *****
Incorrect login, exiting

Tryck ned valfri tangent för att fortsätta...
```

Om inloggningen lyckas ska resultatet bli

```
C:\Windows\system32\cmd.exe
***** PostgreSQL C/C++ interface demo *****

Connected to Postgres DB on localhost

Enter first name and email address for the duck!
Donald d.duck@quacksville.net
Enter password *****
Donald is now logged in.

All persons in Sportsclub8:

*****
Förnamn      Efternamn    Gatadress    Postnr      Postort      Epost
Torbjörn    Karlsson    Fågelvägen 6  11577       Stockholm    tobbbeb@gmail.com
Örjan       Larsson     Svedjegatan 201 11577       Stockholm    orjan@larsson.se
Kalle       Andersson   Storgatan 5a   41871       Göteborg     kalle@andersson.se
Bertil      Eriksson    Havsvägen 22   79331       Leksand      erikssonb@gmail.com
Anna        Holmström   Treviksgatan 43 80627       Gävle        holmsara@hotmail.com
Hanna       Olsson      Prästgatan 15c 83140       Östersund    hannaolsson@hotmail.com
Gertrud     Fredriksson Prästgatan 15b 83140       Östersund    gerfre@live.se
Sara        Larsson     Vintervägen 112 83140       Östersund    larsson.s@gmail.com
Johan       Timren      Statsgatan 22  83141       Östersund    johan.timren@miun.se
Fredrik     Håkansson   Bakutgatan 14d 83141       Östersund    fredan80@gmail.com
Melinda     Jonsson     Lagmansvägen 37 83432       Brunflo      mellis@jonsson.net
Robert      Jonsson     Lagmansvägen 37 83432       Brunflo      robert.jonsson@miun.se
Donald      Duck        Quacker road 13 13131       Quacksville  d.duck@quacksville.net
*****

Tryck ned valfri tangent för att fortsätta...
```

---

För att åstadkomma inmatning av lösenord där inmatade tecken ersätts med \* använder du funktionen `std::string getPassword()` som deklarerats i `PWininput.h` och implementeras i `PWininput.cpp`. Lägg till dessa till projektet.

För att åstadkomma en lösning med korrekta svenska tecken måste konverteringar göras.

Problemet har flera delar:

- Teckenkodningen i databasen är UTF-8 är en längdvarierande teckenkodning som används för att representera text kodad i Unicode som en sekvens av byte (oktetter). Ett tecken kan kodas i upp till 21 bitar i tre bytes. Alla svenska ryms dock inom två bytes.
- C++11 har inget bra stöd för UTF-8, det kommer i C++17.
- För att visa svenska tecken i Windows console tecken ska codepage Windows-1252 användas. Det är en Windows egen utvidgning av ISO 8859-1.

För att lösa dessa problem måste konverteringar göras av strängarna i koden. I filen `winutf8.h` finns två funktioner som hanterar detta.

- `utf8win1252(const char* inCStr, string &outString)` tar en C-sträng i UTF-8 och konverterar till en C++-string kodad i win1252.
- `win1252utf8(const string &inWinStr, string &outUtfString)` tar en C++-string kodad i win1252 och konverterar den till en C++-string kodad i UTF8.

## Redovisning

Uppgiften redovisas med den nya versionen av `DBtest.cpp` där du också kommenterar den kod så du lägger till.