

Mariusz Szafrąński,  
Jan Kubań

# Q-Line 3000 – tworzenie aplikacji bazodanowych

## Na CD:

Na CD znajdziecie wszystkie listingi umieszczone w artykule wraz z programem Q-Szkielet 3000 potrzebnym do tworzenia nowych programów, wiele gotowych przykładów stworzonych przy pomocy technologii Q-Line 3000 oraz pełną dokumentację.

Q-Line 3000 jest technologią wytwarzania oprogramowania opartą o oryginalną koncepcję wykorzystania elementów standardowych – takich dobrze zaprojektowanych klocków, które można ułożyć we własne programy, w zależności od zaistniałych potrzeb. Do funkcjonowania wykorzystuje technologie jawowe, więc programy bez problemów będą działać pod Windows i Linuksem. Programy tworzy się prosto i szybko, a technologia nadaje się zarówno do tworzenia programów jednostanowiskowych, jak i systemów działających w różnego typu sieciach rozległych. Administratorzy takich systemów mogą w bardzo dalekim stopniu modyfikować programy i systemy serii 3000.

Nie ma lepszego sposobu na zapoznanie się z technologią, jak wypróbowanie jej w praktyce. Zachęcamy zatem do samodzielnego zbadania możliwości Q-Line 3000.

## Wykonanie prostego programu

Pierwszym zadaniem jest dodanie do programu tabeli adresowej z możliwością wprowadzania, usuwania i modyfikowania rekordów; szybkiego ich wyszukiwania metodą *spadające literki* i metodą *po całości*; możliwością filtrowania danych i ich zaznaczania; możliwością sporządzania wydruków etykietowych i wydruków wierszowych. Zadanie to można wykonać w trzech krokach: dodamy do menu głównego opcję uruchamiania tabeli adresowej, skonstruujemy samą tabelę, wprowadzimy modyfikacje struktur danych metodą *kamikadze*. Żeby uzmysłowić sobie z jaką technologią macie do czynienia, przed rozpoczęciem realizacji pierwszego zadania konstrukcyjnego włączcie stoper. Po jego wyłączeniu zobaczycie jak wiele otrzymaliście i w jak krótkim czasie!

Dodajemy do menu dodatkową pozycję *Spis adresowy*. W katalogu z programem znajduje się katalog *qcon*. Katalog ten zawiera pliki



Rysunek 1. Logo programów serii Q-Line 3000

ki, w których zapisana jest konstrukcja programu. Cała konstrukcja programu Q-Szkielet 3000 zapisana jest w pliku *menu.qcon*. Należy go odszukać i otworzyć do edycji dowolnym, nieinwazyjnym edytorem tekstowym, takim który nie dopisuje własnych znaków do pliku. Na pierwszy raz zalecamy na przykład *notepad*.

### Opis pliku menu.qcon

W pliku tym za pomocą języka zwanego QCON-em zapisane są wszystkie obiekty programu. Sam plik nie jest zbyt wielki i nawet jeżeli wydaje się skomplikowany postaramy się, by za moment jego treść stała się jasna. Listing 1 opisuje pokrótce elementy w nim występujące.

Plik rozpoczyna się od *//* – są to znaczniki rozpoczynające komentarz. Po nich możemy wpisywać dowolny tekst, interpreter ominie te linie. Kolejna linia to początek bloku definiującego menu główne. Słowo *MENU* oznacza, że definiujemy obiekt typu *MENU*. Słowa *MainMenu* oznaczają, że obiekt ten będzie identyfikowany poprzez identyfikator (nazwę, akronim) *Main Menu*. Interpreter na samym początku poszukuje w plikach *qcon* obiektu *MENU* o nazwie *MainMenu* i od niego rozpoczyna pracę.

Większość obiektów definiowana jest za pomocą listo-macierzy. Tego typu opis obiektów jest bardzo czytelny i bardzo efektywny: można na małej powierzchni zapisać wiele informacji – to taka uwaga na marginesie dla tych, którzy zastanawiają się dlaczego do tego celu nie używamy XML-a.

Po wierszu identyfikującym typ i nazwę obiektu następuje lista specyfikująca atrybuty ogólne obiektu: *TITLE* – nazwa *MENU*. Ciąg znaków umieszczony w " " pojawi się na ekranie. Kolejnym atrybutem jest *POS* – pozycja ekranowa *MENU* podana w procentach. *50 50* oznacza – umieść centralnie. I to koniec listy atrybutów. Dalej w nawiasach { } następuje

Mariusz Szafrąński jest studentem drugiego roku Wydziału Informatyki Wyższej Szkoły Informatyki MILA COLLEGE. Jan Kubań – od niego wszystko się zaczęło.

Kontakt z autorami: [qbs@qbs.com.pl](mailto:qbs@qbs.com.pl)

## Informacje o Q-Line 3000

Technologia Q-Line 3000 została opracowana przez firmę QBS. Pierwsze pomysły dotyczące jej stworzenia pojawiły się w roku 1991. W trakcie pisania programów bazodanowych u różnych klientów zaobserwowano, że problemy wdrożeniowe są bardzo podobne, by nie powiedzieć, że wręcz takie same. Co więcej, dotyczyło to klientów różnych narodowości. Naczelną regułą był fakt, że nigdy program zrobiony według projektu nie zadowalał użytkownika, zawsze po jego instalacji i uruchomieniu pojawiały się prośby o zmianę wydruków, struktur, funkcji itp. Te zmiany to była „pięta achillesowa”. Wymagania użytkowników rosty (tak jak apetyt w miarę jedzenia), a informatycy szybko zniechęcali się ogromem zadań. Trywialne bowiem z pozoru zachcianki pociągały za sobą bardzo żmudne zmiany w konstrukcji programów.

Przy okazji w różnych częściach Europy stwierdzono, że większość problemów można rozwiązać w sposób standardowy – dla przykładu wyszukiwanie w bazie obsługuje się na dwa sposoby: albo poszukiwanie typu *spadające literki* (tak jak w słownikach) albo poszukiwanie podstringu. Zbudowanie mechanizmów ogólnych tak, by dały dołączyć się do każdej bazy rozwiązuje problem na zawsze. Informatycy się nie męczą, a użytkownicy są zadowoleni.

opis macierzowy atrybutów listowych występujących w MENU. Pierwszy wiersz to wiersz identyfikatorów kolumn, kolejne wiersze to wartości kolejnych elementów:

- ID – identyfikator identyfikatorów,
- ACTION – identyfikator akcji wykonywanych po wyborze tej właśnie linii MENU,
- SHORTCUT – skrót klawiszowy (litera) uruchamiający tę akcję. Programy serii 3000 można obsługiwać zarówno myszką oraz z klawiatury,

- NAME – identyfikator atrybutu: nazwa ekranowa danej linii MENU,
- HELP – treść dymków podpowiadających.

Na początku MENU ma tylko dwie pozycje: wywołanie podmenu z podręcznikami i podmenu z operacjami systemowymi. W kolumnie ACTION wywołujemy akcję MENU o identyfikatorze *PodrMenu*. W naszym przypadku *PodrMenu* wywołuje pliki HTML z podręcznikami. Akcja *HTML\_VIEW* służy do wyświetlenia pliku HTML w przeglądarce, natomiast *SRC* to parametr określający ścieżkę do pliku, który zostanie wyświetlony na ekranie. O tym, w jakiej przeglądarce (MS IE lub Mozilla) będzie on wyświetlony, decyduje również konstruktor programu.

*SysMenu* – standardowe menu systemowe, udostępnia ono funkcje systemowe niezbędne w każdym programie (zarówno małym jak i dużym). Za jego pomocą możemy wykonywać na przykład kopie bezpieczeństwa, zarządzać użytkownikami itp. Pomimo, że nigdzie w *menu.qcon* nie definiowaliśmy tych funkcji, są one zdefiniowane w kodzie Java i program automatycznie podłącza odpowiednie pozycje do *SysMenu*.

Znak } zamyka definicję menu głównego oznaczonego identyfikatorem *MainMenu*.

## Dodanie tabeli ADRESY

Po tym krótkim wstępie spróbujmy dodać do menu głównego dodatkową pozycję. Aby to zrobić należy w pliku *menu.qcon* umieścić wpis definiujący nową pozycję w menu. Po tej operacji plik powinien wyglądać tak jak pokazuje to Listing 2.

Słowo *BROWSER* dodane w kolumnie ACTION oznacza, że w tym miejscu wywołujemy obiekt typu *BROWSER* (obsługa lista rekordów danej tabeli) o nazwie *ADRESY\_BR0*. Po wprowadzeniu tej zmiany do pliku *menu.qcon*

**Listing 1.** Definicja menu głównego i menu podręczników

```
//----- DEFINICJA MENU GŁÓWNEGO
MENU MainMenu
  POS = 50 50
  TITLE = "Menu programu"
{
  ID      ACTION      NAME      HELP;
  SYS     MENU SysMenu "Operacje systemowe" "Menu Funkcji Systemowych";
  PODR    MENU PodrMenu "Komplet podręczników" "Menu Podręczników";
}
//----- DEFINICJA MENU PODRĘCZNIKÓW
MENU PodrMenu
  TITLE = "Menu podręczników"
  POS = 50 50
{
  ID      ACTION      NAME      SHORTCUT;
  OGOLNY  HTML_VIEW SRC="podr/og/po.html"; "Podręcznik ogólny" 0;
  PERROR  HTML_VIEW SRC="podr/p_err.html"; "Problemy z drukowaniem" D;
}

```

**Listing 2.** Definicja menu głównego z dodanym spisem adresowym

```

MENU MainMenu
  POS      = 50 50
  TITLE   = "Menu programu"
{
  ID       ACTION          NAME          HELP;
  ADR     BROWSER ADRESY_BRO; "Spis adresowy"  "Obsługa spisu adresowego";
  LIN     LINE              -              -              ;
  SYS     MENU SysMenu     "Operacje systemowe" "Menu Funkcji Systemowych";
  PODR    MENU PodrMenu    "Komplet podręczników" "Menu Podręczników";
}

```

proszę uruchomić program (przez ikonkę na pulpicie lub poprzez plik *q.bat*) – w menu pokazała się nowa pozycja *Spis adresowy*.

Gdy będziemy próbowali w nią wejść program zakończy działanie, ponieważ nie zdefiniowaliśmy obiektu o nazwie *ADRESY\_BRO*, który wywołujemy w menu. Wszystkie błędy występujące podczas projektowania i późniejszej pracy z programem zapisywane są w pliku *out.txt*. Na podstawie wpisów z tego pliku możemy odnajdywać i eliminować błędy. Aby naprawić błąd, który wystąpił należy zdefiniować w pliku *menu.qcon* trzy obiekty:

- tabelę o nazwie *ADRESY*, w której opiszemy strukturę rekordu adresowego;
- obiekt edycji rekordu adresowego o nazwie *ADRES\_EDIT*;
- obiekt wyświetlający spis adresów nazwany *ADRESY\_BRO*.

Listing 3 przedstawia treść definicji tych trzech obiektów. Należy je dołączyć na koniec pliku *menu.qcon*. Dodajmy kilka słów wyjaśnienia poszczególnych pozycji, umieszczonych w tym listingu:

- *TAB* – słowo kluczowe oznaczające obiekt;
- *ADRESY* – identyfikator tabeli i zarazem struktur danych. Słowo *ADRESY* jest wymyślone przez konstruktora. Można wstawić dowolną nazwę dla tworzonej tabeli (zalecamy, by nazwy takie pisać dużymi literami bez użycia polskich znaków oraz spacji);
- *TITLE* – nazwa tabeli dla użytkownika programu (to co będzie wyświetlone na górnej belce okna);
- *BASE* – nazwa zbioru dyskowego tu będzie to *adresy.dat* (także w tym przypadku nazwy zbiorów powinny być pisane dużymi literami bez użycia polskich znaków oraz spacji);

## Elementy standardowe

W rozumieniu Q-Line 3000 elementem standardowym jest dobrze zaprojektowany i rozsądnie sparametryzowany moduł programowy, który rozwiązuje konkretne zagadnienie praktyczne.

Jednym z przykładów takiego elementu jest obsługa spisu (tabeli) z pełnymi możliwościami edycji, modyfikacji, szukania, filtrowania, zaznaczania i wyboru. Te wszystkie możliwości muszą być dostępne od razu po zbudowaniu struktury tabeli, bez względu na to z jakich pól złożone są rekordy tej tabeli.

Podczas opracowywania elementów standardowych przestrzegane są następujące zasady:

- *zasada węzła* – jeżeli element standardowy za coś odpowiada, to nie może być dublowany w innych elementach;
- *zasada inteligentnego interfejsu* – która mówi, że jeżeli element standardowy potrzebuje zewnętrznych informacji, to sam powinien się o nich dowiedzieć. Typowym przykładem może być dołączanie modułu wydruków etykietowych QM-Labels do tabel – wystarczy wskazać modułowi tabelę, a resztę, czyli jej strukturę, typy pól i wzajemne powiązania z innymi tabelami moduł rozpoznaje sam;
- *zasada minimalizacji rykoszetów* – czyli lepiej by element standardowy czegoś nie robił niż ma to robić źle, bo w takim przypadku najczęściej traci się na obsłudze błędów. Niejednokrotnie rezygnuje się z optymalizacji kodu lub zbytniego komplikowania parametryzacji w celu uniknięcia powstania potencjalnych błędów;
- *zasada praktycznego zastosowania* – najlepiej jest, gdy elementy standardowe dobrze rozwiązują typowy problem konstruktora lub użytkownika. Przykładem stosowania tej zasady jest moduł Q-Tenberg – moduł przeznaczony do szybkiego tworzenia niezawodnych wydruków księgowych. Rozwiązuje on takie problemy jak przenoszenie na strony, sumę *folio*, sumę *total*.

W taki sposób stopniowo powstawały elementy takie jak: edycja rekordu, obsługa parametrów, obsługa użytkowników, obsługa drukarek fiskalnych, wydruki etykietowe, wydruki wierszowe, generator raportów, generator stron internetowych, moduł ftp itd.

**Listing 3.** Pełna definicja tabeli ADRESY

```
TAB ADRESY
BASE      = ADRESY
BASETYPE  = DEFAULT
RECEdit   = ADRES_EDIT
ORDERS    =
{
  ID  FIELDS      LABEL      VIS ;
  ID  ( ID )      "Id"        HIDE UNIQUE=YES;
  NAME (LASTNAME ID) "Po nazwisku" SHOW ;
  COMP (COMPANY ID)  "Po firmie"  SHOW ;
}
{
  ID      LEN  TYPE  NAME      ;
  ID      8   INT   "Id" VIS=HIDE ;
  NAME    30  STRING "Imię"   ;
  LASTNAME 40  STRING "Nazwisko" ;
  COMPANY  60  STRING "Firma"   ;
  ZIPCODE  6   STRING "Kod pocztowy" ;
  CITY     30  STRING "Miasto"  ;
  STREET   40  STRING "Ulica"   ;
  PHONE    40  STRING "Telefon"  ;
  MOBPHONE 40  STRING "Telefon kom." ;
  FAX      40  STRING "Fax"     ;
  EMAIL    40  STRING "E-Mail"  ;
}
TAB ADRES_EDIT extends ADRESY;
TITLE   = "Edycja kontaktu"
LAYOUT = VPAN(
  TCP( NAME LASTNAME COMPANY ZIPCODE CITY STREET )
  TCP( PHONE MOBPHONE FAX EMAIL )
)
{;}
TAB ADRESY_BRO extends ADRESY;
TITLE   = "Spis kontaktów"
SIZE    = 0 50
BROWSER_ACTIONS = +FULL_SET +F1MENU
{;}
```

- BASETYPE – rodzaj mechanizmu obsługującego bazy danych, czyli rodzaj używanego silnika baz danych (może to być serwer baz danych Q-SEP lub np. PostgreSQL). Programy serii 3000 mogą obsługiwać równolegle różne bazy w różnych serwerach baz danych;
- RECEdit – wskazanie na obiekt, który będzie odpowiedzialny za edycję rekordu adresowego (ADRESY);
- ORDERS – zdefiniowanie uporządkowań spisu adresowego:
  - ID – identyfikator klucza uporządkowań,
  - FIELDS – w nawiasach lista pól tworząca klucz,
  - LABEL – nazwa uporządkowania dla użytkownika programu,
  - VIS – czy klucz jest widoczny dla użytkownika.

Następnie definiuje się macierz (inaczej można powiedzieć: blok, tabela) definiująca kolejne pola rekordu. W wierszu pierwszym nazwy atrybutów, a w kolejnych poszczególne ich wartości.

W pierwszej kolumnie identyfikator pola ID, w drugiej jego długość LEN, w trzeciej typ TYPE, a w czwartej nazwa dla użytkownika programu – NAME.

Zdefiniowaliśmy kilka pól. Pierwsze to Id, ostatnie E-Mail. Rozmawiając z użytkownikiem programu będziemy mówić właśnie o polu "Id" lub o polu "E-Mail". Natomiast konstruktor programu będzie posługiwał się identyfikatorami (akronimami) tych pól i będzie zlecał na przykład wydruk pola ID .. EMAIL.

Po zdefiniowaniu tabeli głównej musimy określić w jaki sposób będziemy wprowadzać do niej dane. Posłużymy się obiektem ADRES\_EDIT, który określi sposób funkcjonowania i ekranowy wygląd formularza do wprowadzania danych. TAB to słowo kluczowe ADRES\_EDIT. Identyfikator obiektu extends oznacza, że będzie rozszerzał obiekt ADRESY, czyli obiekt ADRES\_EDIT dziedziczy wszystko z obiektu ADRESY. W TITLE podajemy nazwę okna edycji. To, w jaki sposób rozmieścimy pola na ekranie zapisujemy w atrybucie LAYOUT.

Jako ostatnia zdefiniowana jest tabela odpowiadająca za wyświetlanie listy rekordów. Jest to obiekt ADRESY\_BRO. Ten obiekt wywołany jest w menu i w jego definicji określimy jakie akcje będą dostępne na spisie rekordów. Za definicję poszczególnych akcji odpowiada parametr BROWSER\_ACTIONS. Akcje takie jak dodanie, usunięcie, modyfikacja rekordu, wyszukiwanie itp. Podając +FULLSET konstruktor decyduje, by w tym spisie był dostępny pełny zestaw funkcji tego typu. Jeżeli chce, na przykład, usunąć tylko możliwość modyfikacji rekordu, powinien dopisać -MODIFY.

Wykonaliśmy dwa żmudne etapy pracy. Zdefiniowaliśmy podstawową strukturę danych i menu główne, za pomocą którego będziemy się mogli do tych danych dostać. Teraz należy wykonać krok trzeci – *kamikadze!*

Jeżeli podczas procesu tworzenia programu występują błędy lub aplikacja niespodziewanie kończy swoje działanie, to prawdopodobnie w pliku *menu.qcon* zapisaliśmy coś niezgodnie z regułami QCONL-a. Proponujemy podejrzeć plik *out.txt* i tam szukać błędów.



**Rysunek 2.** Start z technologią Q-Line 3000

**Listing 4.** Sposób formatowania pól w tabeli ADRESY

```
{
ID      LEN TYPE  NAME;
ID      8  INT   "Id"      VIS=HIDE;
NAME    30 STRING "Imię"    ATTRIB=(FCAPITAL);
LASTNAME 40 STRING "Nazwisko" ATTRIB=(FCAPITAL);
COMPANY 60 STRING "Firma"   ATTRIB=(FCAPITAL);
ZIPCODE  6  STRING "Kod pocztowy" MASK=
                                MASK("00\\-000");
CITY     30 STRING "Miasto"   ATTRIB=(CAPITAL);
STREET  40 STRING "Ulica"   ATTRIB=(FCAPITAL);
PHONE   40 STRING "Telefon"  ;
MOBPHONE 40 STRING "Telefon kom." ;
FAX     40 STRING "Fax"     ;
EMAIL   40 STRING "E-Mail"  ;
}
```

Po zmianie struktury pierwsze uruchomienie się nie uda, ale proszę się nie przejmować. Za chwilę będzie już lepiej. Wchodzimy do katalogu głównego programu i uruchamiamy *q.bat*. Na ekranie pojawi się duże białe okno aplikacji Q-Line 3000, a po chwili w tym oknie uruchomi się złowrogi komunikat, a po nim kilka następnych (w razie czego proszę sprawdzić na belce zadań okienka *Error* itp.). Trzeba to wszystko pozamykać i niczym się nie przejmować. Program Q-Line 3000 wykonał w tym czasie bardzo pożyteczną pracę: przygotował struktury dla serwera baz danych. Zrobił to na podstawie przygotowanego przez nas zbioru *menu.qcon*. Dlaczego się nie uruchomił? Nie udało mu się to, ponieważ serwer baz danych nie wiedział z jakimi bazami będzie pracował. Wynika to z tego, że tabele klienta mogą się różnić od tych na serwerze. Tuż przed zaprzestaniem pracy program wygenerował plik *struct.qsr* w katalogu głównym. Plik ten należy skopiować do katalogu *dat*. Teraz podczas uruchomienia nie powinno być już żadnych problemów. Plik *struct.qsr* jest plikiem konfiguracyjnym pracę serwera baz danych Q-SEP. W przypadku stosowania innych baz danych, na przykład PostgreSQL lub Oracle, program (w analogiczny sposób) wygenerowałby tego typu pliki właściwe dla tych serwerów.

Proszę wyłączyć stoper i zapisać ile czasu zajęły te wszystkie operacje. Za moment poznamy większość możliwości skonstruowanego przez nas programu adresowego.

Uruchamiamy program Q-Szkielet 3000 i próbujemy:

- wejść do spisu adresowego i dopisać kilka adresów naszych znajomych. Jak na razie to wprowadzanie jest siermiężne, ale poprawimy to w kolejnym kroku konstrukcyjnym;
- można zapoznać się ze standardowymi operacjami dostępnymi na spisie – opisane są one w dokumentacji umieszczonej na CD i spróbować się nimi pobawić;
- klawiszem [F3] zmieniamy uporządkowanie;
- klawiszem [F6] wyszukujemy po nazwisku;

- klawiszem [s] wyszukujemy podciąg znaków;
- można również skomponować widok samego okna przeciągając myszką kolumny lub edytując jego właściwości;
- dostępny jest również wydruk wierszowy z eksportem danych – klawisz [F8];

Już od tego momentu możemy wykonywać kopie bezpieczeństwa naszych danych standardowymi funkcjami dostępnymi w programie.

Tego typu możliwości jest wiele. To o czym mówimy w tej chwili to *szkoła podstawowa*. Przedsmaikiem do *Uniwersytetu* może być uruchomienie skomplikowanego zarządzania danymi poprzez klawisz [F10]

Prawda jak wiele w tak krótko? Inne technologie w tym czasie kończyłyby się instalować lub w lepszym przypadku wypisały na ekranie nieporadne: *hello world*.

Poprawna postać pliku *menu.qcon* podana jest na płytce w katalogu *LEKCJA-1*.

**Formatowanie wprowadzanych danych**

Mamy już działający program. Można dopisywać rekordy adresowe, za pomocą klawiszy strzałkowych przewijać adresy w górę i w dół. Niestety musimy włożyć trochę wysiłku, by nazwiska i imiona wprowadzać z dużej litery, a czasami zapominamy o kresce w kodzie pocztowym. Każdy szanujący się program powinien sobie z tym radzić.

Atrybuty poszczególnym polom nadajemy w naszym pliku konstrukcyjnym *menu.qcon*. Zmieńmy więc fragment definiujący pola w rekordzie adresowym na kod zawarty w Listingu 4.

Jeżeli parametrowi pola ATTRIB przypiszemy wartość (FCAPITAL) oznacza, że chcemy, by wszystkie słowa w tym polu zaczynały się z dużej litery – ten sposób edycji przypisaliśmy polom *Imię* i *Nazwisko*. Jeżeli przypiszemy wartość (CAPITAL) oznacza to, że chcemy, by wszystko w tym polu było pisane dużymi literami – pole CITY.

Rodzajów masek może być bardzo dużo. Na przykład data, czas. Może również zachodzić potrzeba stworzenia własnej maski edycji (tak jak przy polu *Kod pocztowy*) wówczas stosujemy składnię MASK=MASK(<definicja>); W naszym przykładzie pierwsza cyferka "0"

**Listing 5.** Definicja dodatkowego menu z wywołaniem modułu QM-Labels

```
MENU AdresMenu
TITLE = "Wydruki etykietowe"
POS   = 50 50
{
ID     ACTION      NAME;
LABELS LABELS_ARRANGER; "Moduł tworzenia
                                kopert i etykiet";
}
```

**Listing 6.** Wywołanie na spisie menu operacji dodatkowych

```

MENU MainMenu
  POS = 0 10
  TITLE = "Menu programu"
{
ID      ACTION          NAME          ;
ADRESY BROWSER ADRESY_BRO AdresMenu; "Obsługa spisu
        adresowego" ;
LIN     LINE            -              ;
SYS     MENU SysMenu    "Operacje
        systemowe" ;
PODR    MENU PodrMenu   "Komplet
        podręczników";
}

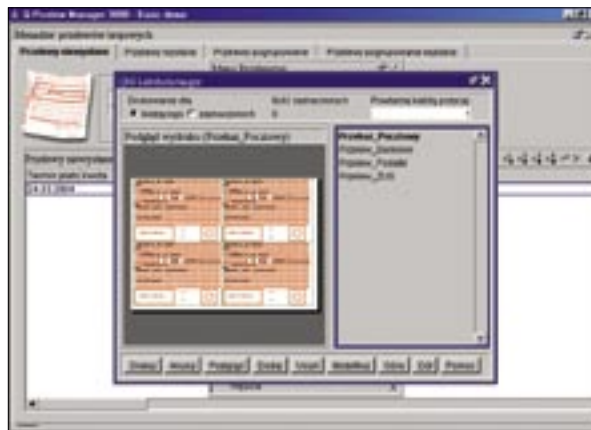
```

oznacza, że pierwszym znakiem pola *Kod pocztowy* może być tylko cyfra, drugim też. Trzeci znak to zawsze "-" (myślnik) i kolejne trzy znaki to cyfry.

Po tych wszystkich zmianach wprowadźmy ponownie dane. Zwróćmy uwagę na to, jak program zapisat pola *Nazwisko* i *Imię*, *Kod pocztowy*. Wszystkie dane program formatuje sam. Proszę wcisnąć klawisz [F10] w celu zatwierdzenia tego rekordu adresowego. Szerzej o możliwościach konfiguracyjnych programów można przeczytać w odpowiednim podręczniku znajdującym się na płycie CD lub stronach QBS.

**Tworzenie prostych wydruków**

Drukowanie w programach serii Q-Line 3000 może odbywać się za pomocą kilku modułów. Jeden z nich – moduł wydruków wierszowych – jest już automatycznie podłączony do spisu adresowego. Poniżej przedstawimy możliwość tworzenia wydruków etykietowych. Odbywa się ono za pomocą jednego z elementów standardo-



**Rysunek 3.** Możliwości modułu QM-Labels na przykładzie programu Q-Przelew Manager 3000

wych: modułu QM-Labels. Jest to element uniwersalny, który można podłączyć do dowolnej bazy (spisu) stworzonej w Q-Line 3000. Dołączenie go do programu odbywa się na dotychczasowych zasadach – przy użyciu języka konstrukcyjnego QCONL. Aby podłączyć moduł do naszego programu należy w pliku *menu.qcon* dodać następujący wpis pokazany w Listingu 5.

W kolumnie *ACTION* pojawiła się nowa wartość *LABELS\_ARRANGER*. Taki wpis spowoduje wywołanie standardowego modułu do tworzenia wydruków. Stworzone menu należy podpiąć do spisu, na którym zamierzamy do używać. W naszym przypadku będzie to spis *ADRESY\_BRO*. Definicja takiej operacji wygląda tak jak na Listingu 6.

Do akcji wywołującej *BROWSER* dodaliśmy nazwę stworzonego przez nas menu funkcji dodatkowych. Interpreter analizując wpis w plikach *qcon* sprawdza, czy po nazwie tabeli istnieje wywołanie dodatkowego menu. Jeżeli znajdzie taki wpis, to podłącza go do tabeli. Po uruchomieniu programu w spisie pojawi się dodatkowa ikona wywołująca menu funkcji dodatkowych. Po wywołaniu dodatkowego menu uzyskamy

**Język konstrukcji – QCONL**

Równoległe do rozwijania elementów standardowych zaistniała potrzeba opracowania metody tworzenia konstrukcji programu, czyli czegoś do łączenia tych elementów w sprawny i użytkowy program komputerowy. Tym czymś jest język QCONL (Q-Line CONstruction Language) – język zapisu konstrukcji programu.

Głównym kryterium, którym kierowano się przy jego opracowywaniu była czytelność. Konstrukcja musi być zapisana w sposób jasny i zwięzły tak, by na pierwszy rzut oka było wiadomo o co chodzi. Najlepszym sposobem okazało się zapisywanie obiektów (elementów standardowych) i powiązań między nimi za pomocą tak zwanej *listo-macierzy*. Każdy element listo-macierzy mógł być również zarazem wskaźnikiem do innego obiektu (innej listo-macierzy).

Listo-macierz to nic skomplikowanego. To po prostu zapis składający się z identyfikatora obiektu oraz listy nazw elementów z podanymi wartościami tych elementów. Po zakończeniu listy następuje zapis macierzy: pierwszy wiersz to wyszczególnienie nazw kolumn, a kolejne to odpowiednie wartości. Listing 1 wyjaśnia na czym polega listo-macierz, ilustruje zapis obiektu MENU.

Wartości zapisane w QCONL-u zwane są *wartościami pierwotnymi*, nadawanymi przez konstruktora programu w odróżnieniu od wartości bieżących i domyślnych, którymi mogą zarządzać użytkownicy. Przykładem może być kolor czcionki. Konstruktor ustawia wartość pierwotną tego koloru na BLACK. Wartość domyślna będzie dziedziczona przez program, ale będzie mogła być zmieniona przez użytkownika na przykład na wartość BLUE. Co więcej użytkownik w niektórych miejscach programu będzie miał możliwość ustawienia własnego koloru na przykład na GREEN.

możliwość tworzenia wydruków w module QM-Labels. Sam proces tworzenia wydruków nie jest zbyt skomplikowany tak więc pozostawiamy czytelnikom.

Po zapisaniu stworzony wydruk jest gotowy do użycia. Można go wywoływać z poziomu menu funkcji dodatkowych, ale jest to czasochłonne i niewygodne w użyciu. Lepszym sposobem na wywołanie naszego wydruku będzie bezpośrednie podpięcie go do spisu. Wystarczy dodać jedną linijkę do pliku *menu.qcon* i stworzony wydruk będzie dostępny po wciśnięciu jednego klawisza. Proszę wpisać kod z Listingu 7.

Dodanie wpisu wywołującego nasz wydruk spowoduje uaktywnienie się ikony z listą wydruków. Ważne jest, aby w parametrze `PRINTNAME="nazwa"` jako *nazwa* podać nazwę, pod którą zapisaliśmy stworzony wcześniej wydruk. Po uruchomieniu aplikacji i wejściu w spis pojawi się nowa ikona. Po jej wybraniu uzyskamy dostęp do listy zdefiniowanych przez nas wydruków. W ten sposób możemy tworzyć i dodawać nowe wydruki do naszej listy, przy czym podpięcie wydruku bezpośrednio do menu jest sprawą dość ryzykowną z punktu widzenia konstrukcji programu. Wyobraźmy sobie na przykład taką sytuację: użytkownik w module QM-Labels stworzy wydruk o nazwie *Adres na kopcercie*, prosi konstruktora o podłączenie tego wydruku bezpośrednio do menu po czym kasuje wydruk w module QM-Labels. Od tego momentu jakiegokolwiek wywołanie wydruku bezpośrednio w menu kończy się niepowodzeniem. Zatem jak widać możliwość jest, ale korzystać z niej należy bardzo ostrożnie.

Poprawna postać pliku *menu.qcon* podana jest w katalogu *LEKCJA-2*.

### Kosmetyka w Q-Line 3000

W tej części artykułu zmienimy nazwę, tapetę oraz ekran wejściowy wyświetlany w programie.

Nazwa, którą będziemy zmieniać jest wyświetlana w lewym górnym rogu programu. Jak widać, jest to tytuł Q-Szkielet 3000, a po znaku "-" opisana jest aktualna wersja programu. Zmieńmy tę nazwę na inną, na przykład Q-Adresownik 3000. Plik, w którym tego możemy dokonać to *version.lic*. Znajduje się on w katalogu głównym programu. Cały opis struktury pliku i zastosowanych w nim poleceń można znaleźć w dokumentacji. Nas interesuje na razie tylko jedna pozycja. Nazwa którą będziemy zmieniać jest ukryta pod zmienną *NAME* i to co jest ujęte w apostrofach zostaje wyświetlone w belce programu. Więc w tym miejscu należy wpisać tekst który nam odpowiada. `NAME = "Q-In-na Nazwa 3000"`; Po wprowadzonych zmianach przy następnym uruchomieniu programu pojawi się już wprowadzona przez nas nazwa.

Gdy mamy już za sobą zmianę nazwy programu zajmijmy się teraz ekranem logowania. Umieszczony tam obraz nie jest zbyt atrakcyjny (są osoby, które boją się tzw. kościotrupów) i jego zmiana zajmie nam tylko chwili-

### Listing 7. Dodanie do menu stworzonych wydruków

```
MENU AdresMenu
  TITLE = "Dodatkowe opcje"
  POS   = 50 50

(
  ID      ACTION          NAME;
  LABELS ARRANGER LABELS_ARRANGER; "Moduł tworzenia
                                     kopert i etykiet";
  WYDRUK LABELS_ARRANGER PRINTNAME="nazwa"; "Wydruk";
)
```

ę. Wszystko co musimy zrobić to utworzyć plik graficzny w formacie *gif*, nazwać go *application.gif* i umieścić w katalogu *pic* naszego programu. Ekran logowania zmieni się automatycznie.

Przejdźmy więc do następnego etapu, by poprzez zmianę tła programu dodać trochę kolorów. W przypadku zmiany tła programu postępujemy nieco inaczej niż w podczas zmiany ekranu logowania. Zmiany wyświetlanej grafiki dokonuje się poprzez modyfikację pliku *qline.txt*, a dokładnie zmiennej *BackgroundImage*. W apostrofach podajemy ścieżkę dostępu do pliku graficznego wraz z jego rozszerzeniem. W tym przypadku *pic* jest nazwą katalogu, a *qbslogo.jpg* nazwą pliku do wyświetlenia. W tym przykładzie należy wgrać plik z grafiką do katalogu *pic* programu Q-Szkielet 3000 i odpowiednio zmodyfikować wpis. Na przykład `BackgroundImage="pic/qbslogo.jpg"`; Zalecamy, aby pliki były przygotowane w formatach *jpeg*, *jpg*, *gif* lub *png*, bo przypadku innych formatów mogą wystąpić problemy z wyświetlaniem. Proszę zwrócić uwagę także na rozmiar pliku, najlepszy do zastosowania w tle naszej aplikacji jest rozmiar 800x600. Jeżeli udało nam się podać właściwą ścieżkę, to powinniśmy zobaczyć wskazany przez nas obrazek. Myślę że możemy być zadowoleni z efektów naszej pracy. Program nabrał trochę kolorów.

## Tworzymy wersję 2.0

Celem tego rozdziału jest stworzenie wersji 2.0 naszego programu, polegającej na dodaniu pola słownikowego (i samego słownika) o nazwie *Imiona* oraz dodanie pola listowego *Lista dzieci*.

### Nowe pola i tabele w programie

Struktura tabeli, którą stworzyliśmy może nie do końca odpowiadać naszym wymaganiom. Pewne wartości, na przykład miasto lub imię, powinny być wybierane ze słowników, a kilku pól brakuje w naszej bazie. W poniższym fragmencie pokażemy w jaki sposób dokonać takich zmian zachowując dotychczas wprowadzone dane. Pierwszą czynnością, jaką należy wykonać, to utworzyć kopię bezpieczeństwa naszych danych. Funkcja, która realizuje te zadanie znajduje się w menu *Operacje Systemowe*. Za pomocą tego modułu możemy wykonywać kopię wszyst-

**Listing 8.** Definicja tabeli IMIONA

```
TAB IMIONA
  TITLE      = "Tabela imion"
  BASE       = IMIONA
  BASETYPE   = DEFAULT
  RECEDIT    = IMIONA_EDIT
  POS        = 40 10
  SIZE       = 36 10
  ORDERS     =
  {
    ID  FIELDS  LABEL;
    ID  ( ID )  "Id" VIS=HIDE UNIQUE=YES;
    IMIE ( IMIE ) "Imię"          UNIQUE=YES;
  }
  {
    ID  LEN  TYPE  NAME          ;
    ID  8    INT   "Id" VIS=HIDE  ;
    IMIE 30  STRING "Imię" ATTRIB=(FCAPITAL);
  }
  TAB IMIONA_EDIT extends IMIONA;
  TITLE      = "Edycja imienia"
  LAYOUT     = TCP( IMIE )
  AER_CONDITION = MBF(IMIE);
  {;}
```

kich lub tylko wybranych tabel zdefiniowanych w programie. W naszym przypadku wykonujemy kopie wszystkich tabel (chcemy zachować wszystkie dane).

Po wykonaniu kopii rozpoczynamy modyfikację pliku *menu.qcon*. Dodamy do naszej struktury dwie bazy MIASTO i IMIONA, które podepnimy jako słowniczki do odpowiednich pól w tabeli ADRESY. Postaramy się stworzyć także jedno pole typu *lista*. Definicję nowych tabel umieszczamy na końcu pliku, tak jak na Listingu 8.

Analogicznie należy postąpić z tabelą MIASTO. Nowe polecenie w kodzie to ustawienie atrybutów ATTRIB=(FCAPITAL). Taki wpis informuje nas o tym, że podczas wprowadzania imienia pierwsza litera będzie wstawiana jako duża. Jeżeli chcemy, aby wszystkie wprowadzane litery były wpisywane dużymi literami, należy zastąpić wpis FCAPITAL słowem CAPITAL. Taki parametr atrybutu możemy zastosować przy wprowadzaniu nazw miast. Pozostałe parametry dla ATTRIB zostały opisane w dokumentacji dołączonej na płycie. Dla pola IMIE ustawiamy atrybut unikalności wpisując w bloku ORDERS przy naszym polu UNIQUE=YES. Taka deklaracja spowoduje, że podczas dodawania imienia nie będzie można wprowadzić dwóch takich samych wpisów – w liście nie pojawi się dwóch Janów. Także nowym atrybutem wprowadzonym do naszego programu to wpis AER\_CONDITION = MBF(IMIE). Taki zapis podczas zatwierdzania rekordu sprawdza warunek, który został podany. W tym przypadku pole IMIE musi zostać wypełnione, w przeciwnym wypadku rekord nie zostanie zapi-

sany do bazy. Taki sam wpis dodajemy przy definicji tabeli MIASTO dla odpowiedniego pola. Podczas definicji tabeli IMIONA\_EDIT został podany sposób, w jakim zostaną wyświetlone poszczególne pola z tabeli. Niestety, w tym przykładzie nie mamy zbyt rozbudowanej struktury pól, dlatego wpis LAYOUT= TCP( IMIE ) wiele nam nie powie. Sposób budowania taba edycyjnego jest skomplikowany i stworzenie ładnie wyglądającego ekranu nie jest prostą sprawą. Dokładniejsze wskazówki można odnaleźć w dokumentacji.

**Podłączenie słowników**

Stworzone tabele pomocnicze IMIONA i MIASTO musimy podpiąć do odpowiednich pól w tabeli ADRESY. Po tej operacji, podczas wprowadzania danych do tabeli ADRESY, przy polach NAME i CITY, będziemy korzystać z danych wprowadzonych do tabel pomocniczych. Aby skorzystać z tej możliwości użyjemy standardowego makra Voc. Wywołanie makra następuje po słowie kluczowym EVH. Wpisujemy nazwę makra, a w nawiasach parametry dla danego niego. Wpis dla pola NAME powinien wyglądać tak: EVH=Voc( IMIONA , IMIE ) ATTRIB=(SUFFLE); Aby nasze makro działało poprawnie, należy w parametrach podać jako pierwszą nazwę tabeli, z której korzystamy, a następnie pole, jakie chcemy pobrać. Dodatkowo, dla parametru ATTRIB pojawiła się nowa wartość. Podanie wartości SUFFLE dla atrybutu powoduje, że wpisywana wartość będzie automatycznie uzupełniana odpowiednim wpisem z tabeli pomocniczej. Po tych modyfikacjach plik *menu.qcon* powinien wyglądać tak jak na Listingu 9.

Spróbujmy uruchomić program. Jeżeli się nie uruchamia, należy sprawdzić plik *out.txt*, a sytuacja powinna się nieco rozjaśnić. Po usunięciu wszystkich

**Listing 9.** Fragment pliku *menu.qcon* definicja tabeli ADRESY z dołączonymi słownikami

```
{
  ID      LEN  TYPE  NAME          ;
  ID      8    INT   "Id"          VIS=HIDE;
  NAME    30  STRING "Imię"          EVH=Voc(IMIONA , IMIE)
  ATTRIB=
  (FCAPITAL SUFFLE);
  LASTNAME 40  STRING "Nazwisko"      ;
  COMPANY  60  STRING "Firma"         ;
  ZIPCODE  6   STRING "Kod pocztowy"  ;
  CITY     30  STRING "Miasto"        EVH=Voc(MIASTO , NAZWA)
  ATTRIB=
  (CAPITAL SUFFLE);
  STREET   40  STRING "Ulica"         ;
  PHONE    40  STRING "Telefon"       ;
  MOBPHONE 40  STRING "Telefon kom."  ;
  FAX      40  STRING "Fax"           ;
  EMAIL    40  STRING "E-Mail"        ;
}
```

błędów związanych ze składnią, program nadal ma kłopoty ze startem. W pliku *out.txt* mamy wpis zaczynający się linijką `qline.sep.SepError: SEP Error: Brak tabeli IMIONA`. Oznacza to niezgodność struktury baz danych między klientem a serwerem. Dodaliśmy pola lub tabele w programie, ale serwer nic o tym nie wie. Aby poinformować serwer baz danych o takich zmianach, należy przekopiować plik *struct.qsr* z katalogu głównego do podkatalogu *dat*. Jeżeli dodawaliśmy nową tabelę, ta operacja wystarczy. W przypadku, gdy dodane zostało pole do istniejącej już tabeli należy także usunąć odpowiedni dla tej tabeli plik z rozszerzeniem *dat* (dlatego na początku wykonaliśmy kopie bezpieczeństwa naszych danych). Teraz uruchomienie programu powinno przebiec bez żadnych niespodzianek, ale zanim przejdziemy do dalszej części, sprawdzimy, czy wszystko działa prawidłowo. Poprawny plik *menu.qcon* zamieściliśmy w podkatalogu *Lekcja-4*.

### Dodanie pola typu LISTA

Kolejnym krokiem w doskonaleniu naszego programu będzie dodanie pola typu *lista predefiniowana* – na przykład pole o nazwie *POWIADOM*, w którym będzie trzymana informacja o sposobie powiadomienia danej osoby. W naszej tabeli *ADRESY* dodajemy kolejne pole, typ tego pola ustawiamy jako *INT*, długość na 8, a jako parametr *NAME* możemy wpisać "Sposób powiadomienia". Teraz to co najważniejsze, czyli definicja samej listy. Definicja listy wygląda następująco `MASK=LLIST( "Nie dotyczy" "Fax" "Mail" "List" "Telefon" )`; lista jest parametrem atrybutu *MASK*, a w nawiasie podane są poszczególne pozycje.

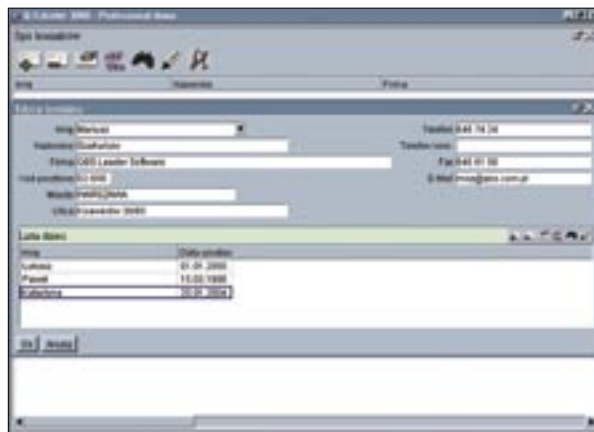
Uruchamiamy program, przechodzimy do spisu adresów i podczas edycji lub dodawania rekordu, pól, które dodawaliśmy, nie widać na ekranie. Aby pole zostało wyświetlone, należy dodać je do *LAYOUT* tabeli *ADRESY\_EDIT*. Nazwę nowego pola wpisujemy w nawiasie za polem *EMAIL*. Teraz po uruchomieniu programu nowe pole powinno być widoczne w table edycyjnym.

#### Listing 10. Plik *data\_versions.cfg*

```

VERSIONNO = 2
VERSIONS =
{
  VER_2 =
  {
    TABLE ADRESY
    {
      ADD POWIADOM INT;
    }
  }
  VER_1 =
  {}
}

```



Rysunek 4. Ogromne możliwości po kilku minutach pracy

### Zmiana struktury baz danych

Możemy zająć się teraz importem danych z kopii bezpieczeństwa, którą wcześniej wykonaliśmy. Ponieważ została zmieniona struktura baz danych, tabele obecnie znajdujące się na serwerze nie odpowiadają strukturze jaka została zapisana do pliku z kopią bezpieczeństwa. W tej sytuacji przed rozpoczęciem importu należy w pliku *data\_versions.cfg* podać listę wszystkich zmian, jakie zaszły od wykonania ostatniej kopii bezpieczeństwa. Listing 10 przedstawia przykładowy wygląd tego pliku.

Krótkie wyjaśnienie przedstawionego Listingu 10. Pierwsza linia `VERSIONNO=2` informuje nas o numerze aktualnej wersji. Z takim numerem wykonują się pliki kopii bezpieczeństwa i przy imporcie z kopii ten numer jest sprawdzany. W zależności od numeru aktualnej wersji, podczas importu, moduł zarządzający kopiami przechodzi do odpowiedniego bloku w pliku i sprawdza, jakie zmiany wystąpiły w tej wersji. Definicja poszczególnych bloków rozpoczyna się wpisem `VER_2 =`, a w `{ }` podajemy nazwy tabel i listę zmian jakie nastąpiły. W naszym przykładzie tabela *ADRESY* wzbogaciła się o nowe pole o typie *INT*. Oczywiście, oprócz dodawania pól `ADD` możemy usuwać, zmieniać długość czy typ, a nawet przypisywać konkretne wartości dla poszczególnych pól. W ten oto sposób pokazaliśmy jak bez utraty danych można modyfikować strukturę bazy.

### Dodajemy listę dzieci

Wykonanie tego zadania należy rozpocząć od dodania nowej tabeli. Nazwijmy ją *DZIECI*. Tabela zawierać będzie trzy pola *IMIE*, *DATAUR* oraz *ID\_R*. To ostatnie pole będzie łącznikiem między tabelą *DZIECI* a tabelą *ADRESY*. Podczas definiowania tabeli *DZIECI* należy dobrać uporządkowanie o takiej samej nazwie, jak pole po którym łączymy. W tym przykładzie będzie to pole *ID\_R*. Listing 11 przedstawia fragment kodu jaki należy wpisać przy definicji tabeli.

Tak jak w przypadku pola *NAME* w tabeli *ADRESY* i tu wykorzystamy słownik imion i podepnijemy go do odpowiedniego pola. Na pole *DATAUR* nakładamy maskę *DATE*, a parametr (`CALENDAR=TRUE`) oznacza podłączenie

**Listing 11.** Definicja tabeli DZIECI

```
TAB DZIECI
BASE      = DZIECI
BASETYPE  = DEFAULT
SIZE      = 0 50
RECEdit   = DZIECI_EDIT
ORDERS    = (
  ID       FIELDS LABEL VIS      ;
  ID      ( ID )  "Id"  HIDE UNIQUE=YES;
  ID_R    ( ID_R ) "Id"  SHOW      ;
  NAME    ( NAME ID) "Imię" SHOW    ;
)
{
  ID      LEN  TYPE  NAME;
  ID_R    8    INT   "Id rodzica" VIS=HIDE;
  ID      8    INT   "Id"          VIS=HIDE;
  NAME    30   STRING "Imię" EVH=Voc(IMIONA , IMIE)
          ATTRIB=(FCAPITAL SUFFLE);
  DATAUR 8    INT   "Data urodzenia" MASK=DATE(CALEN
          DAR=TRUE);
}
TAB DZIECI_EDIT extends DZIECI;
  TITLE      = "Edycja wpisu"
{;}
TAB DZIECI_BRO extends DZIECI;
  TITLE      = "Lista dzieci"
  SIZE       = 120 5
  BROWSER_ACTIONS = +FULL_SET +F10MENU
{;}

```

do pola kalendarza, za pomocą którego w łatwy sposób można wybrać odpowiednią datę. Struktura została stworzona. Teraz należy zadbać o prawidłowe wyświetlenie stworzonej listy. Do tabeli ADRESY dodajemy kolejne pole, ale nie w tabeli głównej tylko w tabeli ADRESY\_EDIT, która odpowiada za edycję rekordu. Nowo dodane pole powinno być typu STRING o długości zero i atrybutowi MASK=BROWSER, by wszystko zadziało prawidłowo użyjemy makra ListID, które ustawia łącznik pomiędzy tabelą, na której zostało wywołane, a tabelą podaną w parametrach makra. Definicja takiego makra będzie wyglądać następująco EVH=ListID(DZIECI\_BROW, ID\_R). Ostatnią czynnością jaką należy wykonać, to wyświetlenie nowo dodanego pola, a cały wpis w tabeli ADRESY\_EDIT wygląda tak jak na Listingu 12.

W ten sposób dodaliśmy do naszego spisu osób informację przechowywaną w postaci listy, w tym przypadku jest to lista dzieci, ale równie dobrze może to być lista pozycji na fakturze czy lista przepisanych leków na recepcie. Zastosowań tego typu rozwiązania jest bardzo dużo. Jeżeli wcześniej tego nie zrobiliśmy, to czas najwyższy uruchomić program i przetestować działanie dodanego spisu. Oczywiście, podczas pierwszego uruchomienia należy zastosować procedurę

kamikadze obowiązującą podczas zmiany struktury bazy danych.

## Stawiamy większe wymagania

Celem tego rozdziału jest pokazanie jak można programy serii 3000 uruchamiać w sieci komputerów oraz jak można uruchamiać je pod Linuksami.

### Uruchomienie programu w systemie Linux

W tym celu potrzebna będzie nam maszyna wirtualna Javy w wersji 1.4.1 lub nowszej. Maszyna wirtualna powinna być zainstalowana w systemie tak, aby można było ją wywołać z linii komend, wpisując w konsoli polecenie `java`. Drugą możliwością jest umieszczenie maszyny Javy w katalogu domowym użytkownika w podkatalogu `jre`, na przykład `/home/nazwa_uzytkownika/jre`. Zmodyfikowany przez nas Q-Szkielec 3000 należy przenieść do systemu Linux (bez maszyny wirtualnej dla Windows). Można go w tym celu skompresować lub przegrać bezpośrednio w miejsce docelowe. Tak przygotowana wersja wymaga kilku poprawek.

W pliku `qline.txt` w rekordzie SEP zmieniamy wpis `Host = "seplib"` na `Host = "sepio"` co oznacza, że będziemy korzystać ze specjalnej wersji serwera bazodanowego dla systemu Linux. Należy jeszcze pamiętać o sprawdzeniu ustawień domyślnej przeglądarki HTML. W tym samym pliku odszukujemy wpis `DefaultHTMLViewer` i podajemy nazwę przeglądarki zainstalowanej w naszym systemie, która ma być domyślnie wywoływana. Nowy wpis może wyglądać następująco `DefaultHTMLViewer = "mozilla"`. Ponieważ pomiędzy systemami występują różnice w zapisie separatorów ścieżki, należy w pliku `config.qsr` przy podawaniu ścieżki do pliku zawierającego opis struktury `StructFilename 'DAT/struct.qsr'` oraz pliku przechowującego wartości liczników dla tabel i transakcji `AutoIncFilename`

**Listing 12.** Definicja tabeli DZIECI\_EDIT dodane pole LISTA

```
TAB ADRES_EDIT extends ADRESY;
  TITLE      = "Edycja kontaktu"
  LAYOUT     = HPAN(
    VPAN(
      TCP( NAME LASTNAME COMPANY ZIPCODE CITY
          STREET )
      TCP( PHONE MOBPHONE FAX EMAIL )
    )
    TCP( LISTA )
  )
{
  ID      LEN  TYPE  ;
  LISTA  0    STRING  MASK=BROWSER EVH=
          ListID(DZIECI_BRO, ID_R);
}

```

me 'DAT/autoinc.qsr', zamienić separatory na właściwe dla systemu Linux.

Należy też upewnić się, czy skrypt *q* oraz plik *sepio* mają atrybuty wykonywalności. Jeśli nie, to nadajemy im atrybut wykonywalności wykonując na przykład następującą komendę `chmod +x q sepio`.

Aby uruchomić aplikację należy wykonać skrypt o nazwie *q*. Znajduje się on w katalogu głównym aplikacji. Skrypt ten uruchamia aplikację przy użyciu maszyny wirtualnej dostępnej z linii komend. Jeśli to się nie powiedzie, szuka maszyny w katalogu użytkownika.

Przy pierwszym uruchomieniu program generuje plik ze strukturą danych dla systemu Linux i kończy swoje działanie. Należy więc plik o nazwie *struct.qsr*, który znajduje się w katalogu głównym aplikacji, przegrać do podkatalogu *DAT* i zastąpić istniejący już tam plik o tej nazwie. Ponowne wykonanie skryptu *q* uruchomi aplikację.

### Uruchomienie programu w sieci komputerów

Programy serii 3000 działają w komunikacji klient-serwer. W takim przypadku instalacja w sieci nie zależy od platformy sieciowej – potrzebne są tylko karty i kable łączące komputery oraz zwykła adresacja TCP/IP. Serwery mogą być różne: Q-SEP lub PostgreSQL. My, w celach ilustracyjnych i ze względu na prostotę instalacji, posłużymy się najlepszym polskim serwerem baz danych Q-SEP.

Serwer ten wykorzystywany jest w dwojaki sposób:

- jako wbudowany element programów jednostanowiskowych,
- jako serwer baz danych, czyli niezależny program obsługujący dowolną liczbę programów jednostanowiskowych (klientów).

Programy jednostanowiskowe – instalacja dokonywana jest automatycznie wraz z instalacją programu. Przy czym Q-SEP umieszczony jest w katalogu programu. Natomiast jego uruchomienie odbywa się automatycznie po uruchomieniu programu.

## Instalacja programu Q-Szkielet 3000

Na krążku dołączonym do czasopisma zamieściliśmy program Q-Szkielet 3000, przeznaczony dla informatycznych majsterkowiczów w dwóch wersjach: linuxowej i windowsowej. Instalacja jest szybka, prosta (intuicyjny program instalacyjny) i bezpieczna – jest nieinwazyjna, czyli nie ingeruje w rejestry systemowe i nie zmienia nic w komputerze.

Użytkownik ma możliwość zmiany proponowanego katalogu instalacji, może wyłączyć opcję tworzenia ikony na pulpicie oraz dodania pozycji do menu *Start*. Po zakończeniu instalacji program Q-Szkielet 3000 jest gotowy do pracy.

W celu uruchomienia programu Q-Szkielet 3000 należy kliknąć na znajdującą się na pulpicie ikonę, która została utworzona podczas instalacji. Program można także uruchomić z katalogu, w którym został zainstalowany. Standardowo jest to katalog *C:/Program Files/QBS/Q-Szkielet-3000*. Znajduje się tam plik *q.bat*, za pomocą którego uruchamiamy aplikację.

Po uruchomieniu pojawi się ekran wejściowy z "sympatycznym szkieletem". Jest to *winiетка logowania*, wyświetlana przez element standardowy obsługi użytkowników zwany QM-Users. Obrazek użyty w winietce ilustruje proces konstruowania programów w technologii Q-Line 3000, polegający na obudowywaniu szkieletu konstrukcyjnego. Kliknięcie na przycisk *Rozpocznij pracę* wywoła menu główne programu. Na samym początku menu nie wygląda zbyt okazale, ale to tylko kwestia czasu i odpowiedniej konfiguracji.

### Listing 13. Wpis w pliku *qline.txt* definiujący sposób podłączenia do serwera

```
Sep = record {
    Host = "adres IP serwera"
    Port = 2507;
};
```

Programy sieciowe – instalacja dokonywana jest w sposób automatyczny podczas instalacji programu. Przy czym sam program i Q-SEP umieszczane są w dwóch różnych katalogach. W razie konieczności przeinstalowania serwera Q-SEP na inny komputer, należy wyłączyć go i następnie przekopiować katalog na komputer docelowy.

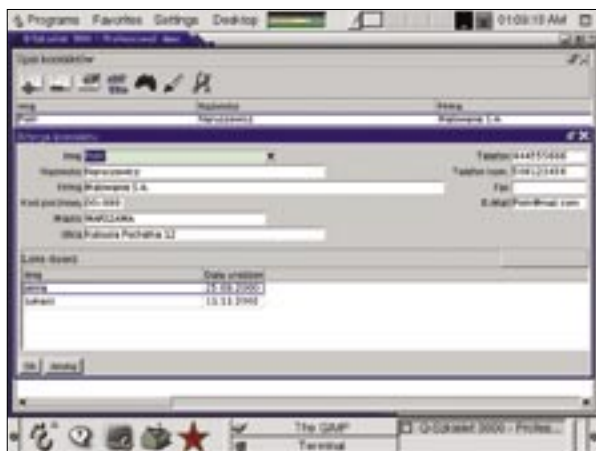
### Instalacja serwera baz danych Q-SEP

Wersja instalacyjna serwera Q-SEP znajduje się na krążku dołączonym do czasopisma. Uruchamiamy plik *exe*, podajemy katalog docelowy i instalujemy. Tak jak w przypadku instalacji klienta, instalacja jest nieinwazyjna. Łączenie z naszym serwerem będzie się odbywać za pomocą zestawu: adresu IP serwera (można podać nazwę) oraz numeru portu, na którym nasłuchuje serwer.

### Startujemy

Przed uruchomieniem serwera należy (tak samo jak w przypadku instalacji jednostanowiskowej) wskazać, jakie struktury baz danych mają być stworzone na serwerze. Plik *struct.qsr* z katalogu głównego programu Q-Szkielet 3000 (klienta) przegrywamy do katalogu *dat* na serwerze. Teraz możemy uruchomić serwer (plik *sep.exe*), pojawi się okienko pokazujące stan pracy serwera (port na którym nasłuchuje serwer, liczbę zalogowanych użytkowników).

Serwer już działa, teraz zajmijmy się klientem. Pierwsza czynność to zadbanie o dane, które wprowadziliśmy



Rysunek 5. A tak wygląda to pod linuxem (Mandrake 9.1)

do tej pory, najlepszym sposobem na ich przeniesienie jest wykonanie kopii bezpieczeństwa z poziomu programu. Następnie edytujemy plik *qline.txt*, a dokładnie jeden wpis w tym pliku pokazany w Listingu 13.

- Host – tu należy wpisać nazwę lub adres IP komputera na którym znajduje się Q-SEP. Domyślnie jest to *seplib* – co oznacza komputer, na którym uruchamiany jest program jednoinstanowiskowy;

- Port – port, na którym nasłuchuje SEP (domyślnie 2507 – patrz plik *config.qsr* w katalogu z serwerem).

Osiągnięty rezultat: potrafimy uruchamiać programy serii 3000 pod Linuxem oraz zainstalować program w sieci komputerów.

## Co dalej?

Zapewne podczas czytania niniejszego artykułu i podczas wykonywania proponowanych przez nas kroków konstrukcyjnych pojawiało się mnóstwo pytań i wątpliwości. Prawdopodobnie nie wszystko udawało się tak gładko jak to opisywaliśmy. Jednakże o możliwościach tej technologii świadczą następujące fakty: dzięki Q-Line 3000 oferta programów z półki firmy QBS jest bardzo szeroka, istnieje sporo programów i systemów działających pod Linuxem, dostępne są praktycznie wszystkie programy do zarządzania firmą i wciąż powstają nowe. Ciekawym przykładem szybkości tworzenia w Q-Line 3000 jest to, że w ciągu zaledwie dziesięciu dni powstał bardzo duży system obiegu dokumentów dla jednej z największych polskich instytucji. Prace rozwojowe trwają nadal. Dla zainteresowanych konstruktorów oprogramowania i dla wdrożeniowców prowadzone są odpowiednie warsztaty tematyczne. ■

R

E

K

L

A

M

A

[www.software.com.pl/konferencje](http://www.software.com.pl/konferencje)

# Bezpieczna Sieć?



To jest możliwe.

Aby skutecznie zabezpieczyć się przed agresorem należy przede wszystkim poznać metody, jakie on stosuje. Zapraszamy na warsztaty, które przybliżą uczestnikom metody stosowane przez komputerowych włamywaczy i w efekcie pomogą obmyślić strategię obrony.

**Techniki przelamywania zabezpieczeń systemów komputerowych (16 edycja)**

11-12 maja 2004, Warszawa

**Techniki ataków na aplikacje i maskowanie w środowisku Windows (9 edycja)**

17-18 czerwca 2004, Warszawa

organizator:

**software**  
**KONFERENCJE**

tel. (022) 860 17 22  
fax. (022) 860 17 71  
[konferencje@software.com.pl](mailto:konferencje@software.com.pl)