POLITECHNIKA WARSZAWSKA
Wydział Elektroniki i Technik Informacyjnych
Instytut Informatyki
Wieczorowe Studia Zawodowe – Informatyka

Rok akademicki 2002/2003

PRACA DYPLOMOWA INŻYNIERSKA

Marcin Stolarski

Notatnik ankietera

Kierownik pracy:
Prof. dr hab. Jan Zabrodzki

Ocena.....

.....
Podpis Przewodniczącego
Komisji Egzaminu Dyplomowego

Kierunek: Informatyka
Imię i nazwisko: Marcin Stolarski
Data urodzenia 20.07.1976 r.
Data rozpoczęcia studiów: 2.02.1999 r.

Ż Y C I O R Y S

Urodziłem się 20 lipca 1976 roku w Warszawie. W latach 1983-1991 uczęszczałem do Szkoły Podstawowej Nr 164 im. Krajowej Rady Narodowej w Warszawie. W roku 1991 rozpocząłem naukę w III L.O. im. Gen. Sowińskiego w Warszawie w klasie o profilu matematyczno-fizycznym. W 1995 roku ukończyłem szkołę średnią zdając egzamin dojrzałości. W tym samym roku podałem naukę na dziennych studiach magisterskich na Politechnice Warszawskiej na wydziale Elektroniki i Technik Informacyjnych na kierunku Informatyka. W 1999 roku przerwałem studia dzienne i rozpocząłem naukę na Wieczorowych Studiach Zawodowych na Politechnice Warszawskiej na kierunku Informatyka oraz podjąłem pracę w firmie Poczta Polska jako Administrator systemów komputerowych.

.....
Podpis studenta

EGZAMIN DYPLOMOWY
Złożył egzamin dyplomowy w dn. .....
z wynikiem .....
Ogólny wynik studiów .....
Dodatkowe wnioski i uwagi Komisji .....
.....
.....

STRESZCZENIE

Gromadzenie danych, tworzenie baz danych, sortowanie danych, generowanie raportów to zadania jakie są robione na rynku biznesowym. Rynek ten ze względu na konkurencję dąży do coraz szybszej realizacji tych zadań i aby temu podołać do tego celu wykorzystuje komputery. O ile przechowywanie baz danych, generowanie raportów, sortowanie danych jest już dość dobrze opanowane, to komputerowe zbieranie danych jest raczej młodą dziedziną. W ramach pracy inżynierskiej napisałem aplikację do zbierania danych (ankiet ulicznych) na przenośny komputer z systemem PalmOS jako że jest to najpopularniejszy, a zarazem jeden z najtańszych palmptopów na rynku.

Praca ta składa się z dwóch części. Pierwsza z nich to analiza problemu i założenia wstępne jakie stworzyłem przy projektowaniu aplikacji. Opowiem też o rozwiązaniach dostępnych na rynku, jak i o samym palmptopie i jego możliwościach.

W drugiej części opowiem o samym projekcie. Można będzie się dowiedzieć jakie zmiany poczyniłem wgłębiając się możliwości jakie daje mi urządzenie. Będzie też o samym środowisku programistycznym jak i o samej strukturze programowania aplikacji PalmOS oraz jakie miałem problemy pisząc aplikacje.

POLLSTER'S NOTEBOOK

Summary

Collecting the data, sorting it and generating reports – these are the tasks in the business. This business, due to the competition, tends to fulfill those tasks faster and faster and in order to achieve this uses computers. Though storing the data, generating the reports and sorting the data are already well managed, collecting the data with the use of computer is a relatively new area. I suggest creating an application for collecting the data (street inquieries) for a portable computer with PalmOS system since it is the most popular and, at the same time, one of the cheapest palmtops available on the market.

The paper will consist of two parts. The first part includes an analysis of the problem and initial assumption which I have taken projecting the application. I will also describe solutions available on the marke and the palmtop itself with its possibilities.

In the second part of the paper I will describe the project itself. I will describe the changes made as a result of deep analysis of the possibilities of this appliance. I will also include detailed information about Development System and the structure of programming PalmOS application and the troubles which arised during the process of creating it.

Spis Treści:

Wstęp

Gromadzenie danych, tworzenie baz danych, sortowanie danych, generowanie raportów to zadania jakie dzieją się na rynku biznesowym. Rynek ten ze względu na konkurencję dąży do coraz szybszej realizacji tych zadań i aby temu podołać wykorzystuje do tego celu komputery. O ile przechowywanie baz danych, generowanie raportów, sortowanie danych jest już dość dobrze opanowane, to komputerowe zbieranie danych jest raczej młodą dziedziną. Proponuje napisanie aplikacji do zbierania danych (ankiet ulicznych) na przenośny komputer z systemem PalmOS jako że jest to najpopularniejszy, a zarazem jeden z najtańszych palmptopów na rynku.

Na rynku ankiety przeprowadza się zwykle w formie testu. Test ten jest dawany osobie ankietowanej, bądź ankieter osobiście zadaje pytania. Do pytania dodawane są proponowane odpowiedzi, które trzeba zaznaczyć w postaci wiele z wielu (checkboxy) bądź jedną z wielu (trigery, listy wyboru). Często ankietowanemu pozostawia się jeszcze możliwość dodania własnej odpowiedzi (dodatkowe pole tekstowe), bądź nie proponuje się odpowiedzi i pozostawia się wolne pole do wpisania. Ze względu na późniejszą analizę komputerową najlepszą formą są gotowe odpowiedzi, które pozwalają niemalże na natychmiastową analizę i generowanie statystyk. Dodatkowo taka forma pozwala na szybkie przepisanie ankiety na komputerze przy wykorzystaniu odpowiednich formatek. Dlatego aplikacja Notatnik Ankietera jest nastawiona właśnie na przeprowadzanie tego typu ankiet. Pozwala także na własne odpowiedzi ankietowanego, ale trzeba się liczyć z tym, że może być to co najwyżej parę wyrazów.

Praca ta składa się z dwóch części. Pierwsza z nich to analiza problemu i założenia wstępne jakie stworzyłem przy projektowaniu aplikacji. Opowiem też o rozwiązaniach dostępnych na rynku, jak i o samym palmptopie i jego możliwościach.

W drugiej części opowiem o samym projekcie. Można będzie się dowiedzieć jakie zmiany poczyniłem wgłębiając się możliwości jakie daje mi urządzenie. Będzie też o samym środowisku programistycznym jak i o samej strukturze programowania aplikacji PalmOS oraz jakie miałem problemy pisząc aplikacje.

Założenia Projektowe

Analiza problemu

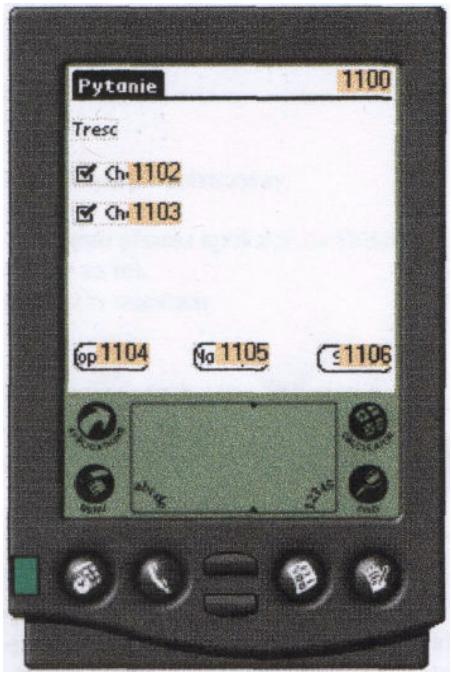
Jak już wspomniałem gromadzenie danych, tworzenie baz danych, sortowanie danych, generowanie raportów to zadania jakie są robione na rynku biznesowym. Rynek ten ze względu na konkurencję dąży do coraz szybszej realizacji tych zadań i aby temu podołać wykorzystuje do tego celu komputery. O ile przechowywanie baz danych, generowanie raportów, sortowanie danych jest już dość dobrze opanowane, to komputerowe zbieranie danych jest raczej młodą dziedziną. Dotychczas widziałem rozwiązania typu papierowych ankiet ulicznych, które są mozolnie przepisywane do komputerów. Rozwiązanie to generuje błędy, oraz nie zabezpiecza przed nieuczciwością ankieterów. Przy wykorzystaniu palmptopa, dane można poddać analizie zaraz po przegraniu ich z urządzenia przenośnego do stacjonarnej bazy danych.

Wstępny projekt programowy

Aplikacja zostanie napisana na palmptopa z systemem operacyjnym PalmOS. Projekt będzie zawierał program na palmptopa, oraz aplikacje na PC konwertującą pliki tekstowe z ankietą na format PalmOS. Do całości zostanie dołączona darmowa aplikacja pozwalająca na zamianę plików baz danych z PalmOS na pliki Excela. Celem budowy urządzenia jest przyspieszenie procesu przeprowadzania ankiety, eliminacje błędów oraz fałszywych wpisów (wykorzystując automatyczną rejestrację godziny systemowej) przy wypełnianiu i przepisywaniu ankiet. Urządzenie typu palmptop z PalmOS (np. Palm 100m) zostało wybrane ze względu na: niską cenę, popularność, dostępność oraz długi czas pracy na jednym komplecie baterii bądź akumulatorów (około 30h ciągłej pracy co przekłada się na 2 tygodnie zwykłego używania).

Aplikacja „Notatnik ankietera" na PalmOS:

Aplikacja będzie zadawała po kolei pytania, zgodnie ze wcześniej wprowadzonym schematem (na podstawie pliku tekstowego). Ankieter będzie widział pytanie [Rysunek 1], oraz możliwe formy odpowiedzi do zaznaczenia (checkboxy, rozwijane menu) lub wpisania (pole typu tekst), oraz przyciski: następny, poprzedni, stop. Wymiana danych z PC odbywać się będzie przy pomocy oryginalnej aplikacji firmy Palm zwanej HotSync. Proces ten zwany zwykle jest synchronizacją i polega na automatycznym przegrywaniu plików miedzy palmptopem a komputerem PC tak aby po obu stronach były identyczne kopie.



Rysunek 1. Przykład projektowanej aplikacji.

Przed rozpoczęciem ankiety trzeba będzie ją przygotować w formie pliku tekstowego, który zostanie następnie skonwertowany na formę zrozumiałą dla systemu PalmOS oraz dla samej aplikacji. Aplikacja do konwersji będzie rozpoznawać następujące początki linii:

komentarz
@# komentarz

pytanie
@q pytanie

checkbox
@c checkbox odpowiedz

triger
@t triger
odpowiedz

lista wyboru
@ lista
odpl
odp2
odp3
@k

odpowiedz tekstowa
@x tekst

koniec pytania
@a koniec pytania

Przykładowa fraza pytania będzie miała następującą postać:

@# Przykładowe pytanie
@qJak często piszesz aplikacje na PalmOS?
@cRaz na rok
raz na rok
@cRaz w miesiącu
raz w miesiącu
@xInne:
@a

Na palmptopie zobaczymy:
Jak często piszesz aplikacje na PalmOS?
() Raz na rok
(*) Raz w miesiącu
Inne:.....

A w bazie danych zostanie zarejestrowane:

Nr rekordu	Nazwa użytkownika	Numer ankietowanego	Data ankiety	Godzina ankiety	Pytanie	Odpowiedź
1	Marcin Stolarski	234	23.10.2001	00.39	Jak często piszesz aplikacje na PalmOS?	raz w miesiącu

Fraza składa się z:
- pytania
- podpowiadanej odpowiedzi (wiele checkbox-ów lub wiele trigerów lub jedna lista)
- opcjonalnie z dodatkowej odpowiedzi (tekst)
- końca pytania

Jeżeli wypełnimy pole tekst i użyjemy:
- checkbox-ów lub listy, to jako odpowiedź będzie brany tekst.
- trigerów to odpowiedź będzie się składała z zaznaczonych odpowiedzi (włącznie z polem tekst) porozdzielanymi średnikami.
Konwerter ten będzie narzędziem z interfejsem typu wsadowego.

Rozwiązania dostępne na rynku

Na rynku można spotkać inne przenośne urządzenia do gromadzenia danych. W głównej mierze są to przenośne terminale (Casio, Formula, Symbol) wyposażone w skaner kodów kreskowych oraz klawiaturę. Dodatkowo urządzenia te mają podwyższoną odporność na zniszczenie. Wadą ich jest dość wysoki koszt oraz niewygodne klawiatury, które są wykorzystywane w sytuacjach gdy skaner kodów kreskowych nie radzi sobie ze zniszczoną etykietą. Terminale te są głównie przeznaczone do zbierania informacji z etykiet produktów, na których są kody kreskowe i raczej nie nadają się do przeprowadzania ankiet ulicznych.
Kolejną grupą urządzeń jest cala gama produktów Psion, poczynając od komputerków przypominających małe laptopy (np. Psion 5x) a kończąc na przenośnych terminalach, jakie powszechnie wykorzystują inkasenci gazowni. Urządzenia te nadają się do naszych celów, jednak są dość drogie.

Na koniec proponuję się przyjrzeć urządzeniom PalmOS. Są to palmptopy u których miejsce klawiatury zastąpiło pole grafiti, na którym wpisujemy odręcznie litery przy wykorzystaniu specjalnego piórka. Ze względu na dużą popularność na rynku proste modele są tanie (poniżej 100$), a także można spotkać modele ze zwiększoną odpornością na zniszczenie oraz z wbudowanym skanerem kodów kreskowych (Symbol).

Przykłady zastosowań palmptopów PalmOS

Bardzo dobry opis urządzeń z PalmOS oraz ich zastosowań zawarty jest w artykule „Palm, czyli co? Artykuł o podstawach urządzeń Palm i systemu Palm OS. Szczególnie przydatny dla nowych i przyszłych użytkowników" Dominika Oślizło [3].

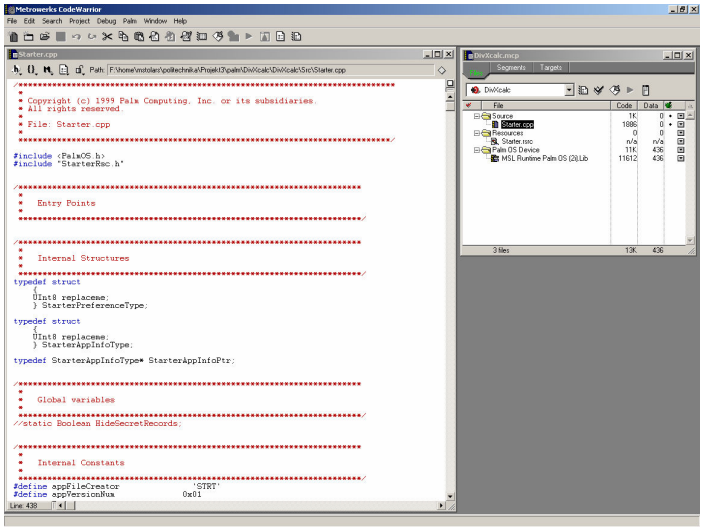
Palmptopy z PalmOS to urządzenia produkowanie głównie przez firmę Palm, ale także przez firmy Sony, Handspring, Symbol. Wszystkie urządzenia wykorzystują procesor Motoroli, z rodziny procesorów 68000 (16-32MHz), pamięć RAM (2-16MB), pamięć ROM/FLASH (2-4MB) na system operacyjny, czarno-biały ekran o rozdzielczości 160x160 (niektóre drogie modele posiadają kolorowy ekran 320x320). Przeciętny czas pracy na komplecie baterii wynosi jeden miesiąc. Palmptopy komunikują się z innymi urządzeniami przy wykorzystaniu portów podczerwieni, portu RS232 bądź portu USB. Niektóre modele posiadają dodatkowy slot na karty rozszerzeń takich jak pamięć flash czy BlueTooth. Powszechnie na rynku wykorzystywane są jako zaawansowane organizery z możliwością dodania dodatkowych aplikacji (np. przeglądarek WWW, programów pocztowych, edytorów kompatybilnych z Word i Excel). Ich najmocniejszymi atutami są mała wielkość i waga, powszechność na rynku, niska cena i dość duża prędkość działania ze względu na doskonałe wykorzystanie zasobów przez system i aplikacje.

Projekt Programowy

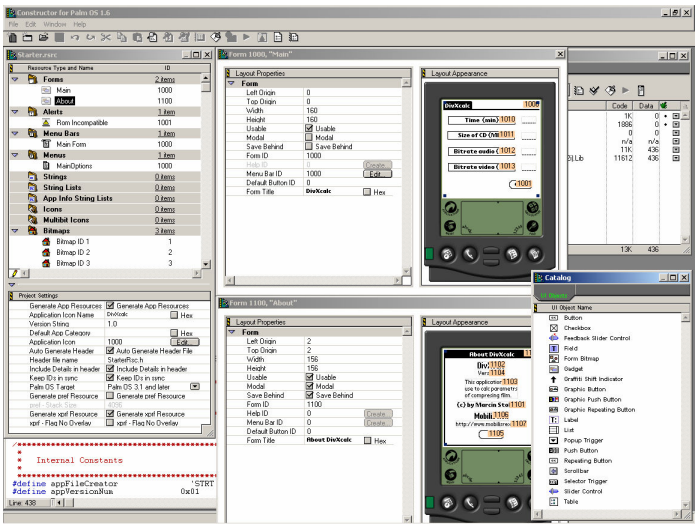
Narzędzia programowe

Generalnie narzędzia programowe można podzielić na dwa rodzaje. Pierwszy rodzaj w głównej mierze oparty jest na licencji GNU, w szczególności chodzi o kompilator GCC na procesor Motorola, oraz pomocnicze narzędzia do projektowania aplikacji, z których część nie jest już niestety darmowa. Całość może pracować pod MS Windows, bądź jakieś środowisko Unix w szczególności LINUX. Szczegółowy opis można znaleźć w artykule Huberta Trzewika pt. „Szybki wstęp do programowania PalmOS" [4].

Drugim rodzajem narzędzia (i to użyłem do napisania aplikacji) jest środowisko programistyczne CodeWarrior firmy Metrowerks [7]. Przypomina ono nieco środowisko Microsoft Visual C++. Składa się z dwóch części. Pierwsza część (CodeWarrior IDE) służy do pisania kodu, i kompilacji [Rysunek 2], druga (Constructor) służy do interaktywnego modelowania interfejsu przyszłej aplikacji [Rysunek 3].



Rysunek 2. CodeWarrior IDE.



Rysunek 3. Constructor.

Przed rozpoczęciem pracy trzeba wpierw zainstalować samo środowisko CW. Potem należy doinstalować stosowne uaktualnienia dostępna na stronach www.palm.com, amianowicie chodzi o sdk 3.5 oraz sdk 4.0, które zawierają dodatków biblioteki pozwalające na korzystanie z systemów PalmOS w wersji 3.5, oraz 4.x. Kiedy już to uczyniliśmy należy skonfigurować nasze środowisko pracy na pisanie aplikacji PalmOS (szczegółu są dostępne w dokumentacji, bądź w książce „PalmOS Programming Bible" [1]. Kolejnym krokiem jest otworzenie nowego projektu (najlepiej Aplikacji PalmOS 3.5). Generowana jest kod pozwalający na podstawową funkcjonalność aplikacji (otwarcie okna głównego i menu help). Pojawia się okno w którym możemy doszukać się plików Starter.cpp zawierający wygenerowany kod (i w tym pliku będziemy pisać dalszy kod naszej aplikacji) oraz Starter.rsrc w którym przy wykorzystaniu modułu Construktor będziemy mogli modelować formatki naszej aplikacji.

Ostatnią aplikacją niezbędną do pisania aplikacji PalmOS jest emulator komputera Palm o nazwie POSE [Rysunek 4]. Korzystanie z niej pozwala na sprawdzenie aplikacji, a w przypadku gdyby aplikacja uszkodziła system operacyjny emulowanego palmptopa, można błyskawicznie przywrócić stan z przed awarii. Dodatkowo emulator ten posiada zaawansowane funkcje debagingu, oraz testowania samej aplikacji np. na przypadkowe wskazania, czy losowe ciągi (opcja gremlins).

Rysunek 4. POSE.

Zasady pisania aplikacji PalmOS

Podstawowe zasady pisania aplikacji dość szczegółowo są opisane w artykule Huberta Trzewika pt. „Szkielet aplikacji PalmOS" [5]. Aby móc dalej rozwijać swoje umiejętności polecam książkę Lonnona R. Fostera pt. „Palm OS Programming Bible" która zawiera wiele przykładów różnych aplikacji. Do książki dołączone jest płyta CD z przykładami oraz z samą książką w postaci pdf. Dostępna jest też w sieci (jako pdf) książka Orielly „Palm Programming - The Developers Guide", ale trzeba się liczyć z tym że najnowsze rozwiązania takie jak obsługa kolorowych ekranów, czy wymiennych kart pamięci nie będą tam opisane.

Aplikacje PalmOS oparte są na generowaniu, przepływowi i obsłudze zdarzeń. Jeśli aplikacja nie obsłuży jakiegoś zdarzenia, jest ono przekazywane do systemu. Wynikiem obsługi jednego zdarzenia może być narodzenie się kolejnych zdarzeń (np. opuszczenie i podniesienie rysika może zaowocować uruchomieniem aplikacji). Uruchomienie aplikacji zaczyna się od funkcji `PilotMain()`. Jako parametry wejściowe dostajemy informacje w jakim trybie aplikacja została uruchomiona (uruchomienie zwykłe, wywołanie funkcji find w celu przeszukania rekordów, uruchomiania aplikacji HotSync, resetu palmptopa, alarmów i innych zdarzeń, o które aplikacja poprosiła menagera systemu). Jak więc widać aplikacje są częściej uruchamiane niż to widać na pierwszy rzut oka i jeśli np. po resecie palmptopa pojawia nam się krytyczny wyjątek, nie jest za niego winny system, ale źle napisana któraś z aplikacji, która nie prawidłowo reaguje na uruchomienie w tym trybie. Jeśli aplikacja została uruchomiona w trybie normalnym, to kolejną funkcją będzie `StartApplication()` odpowiedzialna za przygotowanie samej aplikacji do uruchomienia (np. odtworzenia ustawień z bazy danych aplikacji. Z niej też jest uruchamiana pierwsza formatka (zwykle MainForm). Przed zamknięciem aplikacji wykonywana jest funkcja `StopApplication()` która z kolei zajmuje się przygotowaniem aplikacji do wyłączenia (zwalnianie pamięci, zapisaniem konfiguracji do bazy danych). Dobre

oprogramowanie tych fragmentów pozwoli na zrobienie wrażenia wielowątkowości. Kiedy w menagerze systemu przełączamy się między aplikacjami, faktycznie zamykamu jedna i dopiero uruchamiamy drugą, a wygląd pracującej aplikacji osiągamy przez zapis jej aktualnych ustawień podczas tego zamknięcia. Dalej w kodzie programu znajdziemy funkcje `EventLoop()` która odpowiada za reakcje na wszystkie zdarzenia. Można tam znaleźć rozdzielenie na poszczególne formy przy wykorzystaniu `switch(formID)`.

Końcowy projekt programowy

Założenia

Założenia są podobne do tych z projektu wstępnego. Jedyną zmianą jest to ze do przygotowywania plików z ankietami została wykorzystana aplikacja z pakietu bazy danych MobilDB. Została też zmieniona forma plików tekstowych, jakie będą służyły do generowania pliku z ankietą.

Funkcje programu

Tworzenie ankiety

Ankieta tworzona jest w aplikacji MobilDB, a następnie jest przesyłana do palmptopa, gdzie będzie wykorzystywana przez aplikację Notatnik Ankietera.

Przeprowadzanie ankiety

Ankieter uruchomi aplikacje. Gdy naciśnie przycisk [Rozpocznij ankietę] pojawi mu się pierwsze pytanie. Będzie mógł na nie odpowiedzieć zaznaczając bądź wypełniając stosowne pola. Następnie naciśnie przycisk [>>>] by przejść do kolejnego pytania, bądź przycisk [<<<] by się cofnąć.

Zapis wyników ankiety do bazy danych

Gdy ankieter odpowie na ostatnie pytanie pojawi mu się napis „Koniec ankiety” oraz przycisk [>>>]. Gdy go naciśnie odpowiedzi zostaną zapisane do pliku bazy danych zgodnego z MobilDB.

Exporty bazy danych do pliku tekstowego

Po wypełnieniu ankiety będzie można przegrać plik z odpowiedziami z palma na komputer PC, a następnie za pomocą aplikacji MobilDB wyeksportować wyniki do pliku tekstowego.

Opis implementacji

Organizacja programu

Program składa się z trzech modułów: programowy starter.cpp, oraz moduły związane z zasobami (ikonki, wygląd interfejsu) starter.h i starter.rsrc. Dodatkowo wykorzystuje się aplikacje MobilDB w celu wymiany danych miedzy palmptopem a komputerem PC.

Struktury danych

Dane w programie trzymane są pod trzema postaciami. Typem pierwszym są pytania ankiety w formacie MobilDB. W aplikacji jest ona dostępna jako baza danych (tablica) wypełniona ciągami tekstowymi.

```
//Struktura rekordu bazy
        typedef struct {
                char txt[50];
        } MyRecordType;
```

Ze względu na format MobilDB, pierwsza informacje odczytujemy z komórki 4 (liczba pytań w ankiecie), a następnie odczytujemy rodzaj pytania, pytanie oraz odpowiedzi. Istotne jest to że w ciągach tekstowych trzeba omijać pierwsze osiem znaków, ponieważ są to znaczniki MobilDB.

Kiedy odczytamy z ankiety listę odpowiedzi tworzona jest tabela struktur do wpisywania odpowiedzi, która jest wypełniana w miarę udzielania kolejnych odpowiedzi w ankiecie.

```
//Struktura rekordu odpowiedzi
typedef struct {
        char Lp[10];
        char Owner[50];
        char NrAnk[10];
        char Data[20];
        char Time[20];
        char Pyt[50];
        char Odp[50];
        } MyOdpType;
```

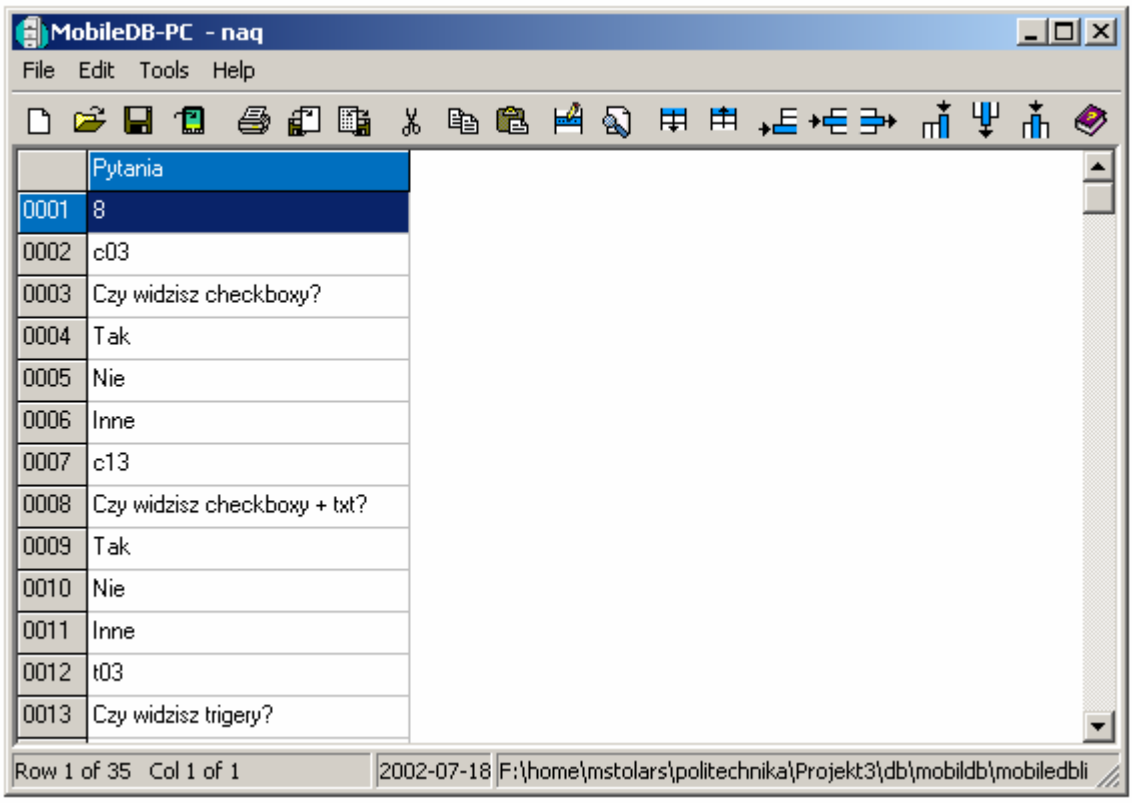
Kiedy osiągamy koniec ankiety, jest ona przepisywana do bazy danych odpowiedzi zgodnie z regułami MobilDB.

```
for (n=0;n<loaptrsize;n++) {
                for(i=0, x=0;x<8;i++,x++) buf[i]=recs[x];
                for(x=0;x<StrLen(loaptr[n]->Lp);i++,x++)  buf[i]=loaptr[n]->Lp[x];  buf[i++]=0;
buf[i++]=1;
                for(x=0;x<StrLen(loaptr[n]->Owner);i++,x++)        buf[i]=loaptr[n]->Owner[x];
buf[i++]=0; buf[i++]=2;
                for(x=0;x<StrLen(loaptr[n]->NrAnk);i++,x++)        buf[i]=loaptr[n]->NrAnk[x];
buf[i++]=0; buf[i++]=3;
                for(x=0;x<StrLen(loaptr[n]->Data);i++,x++)          buf[i]=loaptr[n]->Data[x];
buf[i++]=0; buf[i++]=4;
                for(x=0;x<StrLen(loaptr[n]->Time);i++,x++)          buf[i]=loaptr[n]->Time[x];
buf[i++]=0; buf[i++]=5;
                for(x=0;x<StrLen(loaptr[n]->Pyt);i++,x++)            buf[i]=loaptr[n]->Pyt[x];
buf[i++]=0; buf[i++]=6;
                for(x=0;x<StrLen(loaptr[n]->Odp);i++,x++)            buf[i]=loaptr[n]->Odp[x];
buf[i++]=0;
                buf[i++]=rece;

                //rozmiar rekordu
                UInt16 sizer=i;//8+StrLen(buft)+1+1;

                //dodanie rekordu
                MemHandle memH;
                Char *p;
                UInt16 recindex = dmMaxRecordIndex;
                // Insert code for ADDREC
                memH = DmNewRecord(gLibDBa,&recindex,sizer);
                if (memH == 0)  FrmCustomAlert(InfoAlert,"Failed    to    add    new    record
:(",NULL,NULL);
                        else {
                        p = (Char*)MemHandleLock(memH);
                        if (p == NULL)  FrmCustomAlert(InfoAlert,"Failed    to    lock    down
pointer",NULL,NULL);
                                else {
                                //FrmCustomAlert(InfoAlert,buf,NULL,NULL);
                                MemSet(buf, 0x00, sizer);
                                DmWrite(p, 0, buf, sizer);
                                DmReleaseRecord(gLibDBa,recindex,true);
                                }
                        }
        }
```

Szczegółowy opis formatu MobilDB można znaleźć na dołączonej płycie CD [6].

Opis interfejsu

Opis interfejsu przedstawię jako kompletny proces robienia ankiety. Na początek musimy stworzyć ankietę w standardzie MobilDB. Można robić to na palmptopie, ale zdecydowanie wygodniej jest to robić na komputerze PC. Musi to być baza danych o nazwie naq.pdb.



Rysunek 5. Przygotowanie ankiety w MobilDB.

Jak widać [Rysunek 5] jest to jednokolumnowa tablica ciągów znaków wypełniana zgodnie z następującymi regułami. Pierwszy rekord to liczba pytań. Kolejne rekordy maja strukturę: rodzaj pytania, pytanie, odpowiedzi. Rodzaj pytania składa się z trzech fraz: typu pytania, czy ma być pole tekstowe, ilości odpowiedzi.

Typ pytania oraz czy ma być pole tekstowe	Opis
c0	checkboxy
c1	checkboxy + pole tekstowe
t0	trigery
t1	trigery + pole tekstowe
l0	Lista
l1	lista + pole tekstowe
x0, x1	pole tekstowe

I tak tworząc pytanie w na PC w aplikacji MobilDB w postaci:

```
c13
Czy widzisz checkboxy + txt?
Tak
Nie
Inne
```

Zobaczymy pytanie na ekranie palmptopa [Rysunek 6].

Rysunek 6. Przykładowe pytanie.

Pojawią się trzy odpowiedzi oraz pole txt. Ankietę można przygotować też jako plik txt lub w formacie CSV.

Oto przykład możliwości przygotowania ankiety w formacie txt. Zawiera on wszystkie rodzaje pytań oraz zawiera jeden błąd, w celu testowania aplikacji.

```
{Column Data Types}     Text
Pytania
8
c03
Czy widzisz checkboxy?
Tak
Nie
Inne
c13
Czy widzisz checkboxy + txt?
Tak
Nie
Inne
t03
Czy widzisz trigery?
Tak
Nie
Inne
t13
Czy widzisz trigery + txt?
Tak
Nie
Inne
l03
```

```
Czy widzisz liste?
Tak
Nie
Inne
l13
Czy widzisz liste + txt?
Tak
Nie
Inne
x00
Czy widzisz txt?
a00
Blad
```

Kiedy stworzymy bazę danych przesyłamy ją na palmptopa (można użyć tak zwanej synchronizacji, bądź przesłać jako oddzielny plik), uruchamiamy aplikacje Notatnik Ankietera [Rysunek 7] i rozpoczynamy ankietę [Rysunek 8].

Rysunek 7. Notatnik Ankietera.

Rysunek 8. Ankieta.

W każdej chwili możemy się cofnąć przejść na następnego pytania, bądź przerwać ankietę. Należy jednak pamiętać, ze jeśli się cofamy to tracone są wycofane odpowiedzi i trzeba udzielać jeszcze raz odpowiedzi na kolejne pytania.

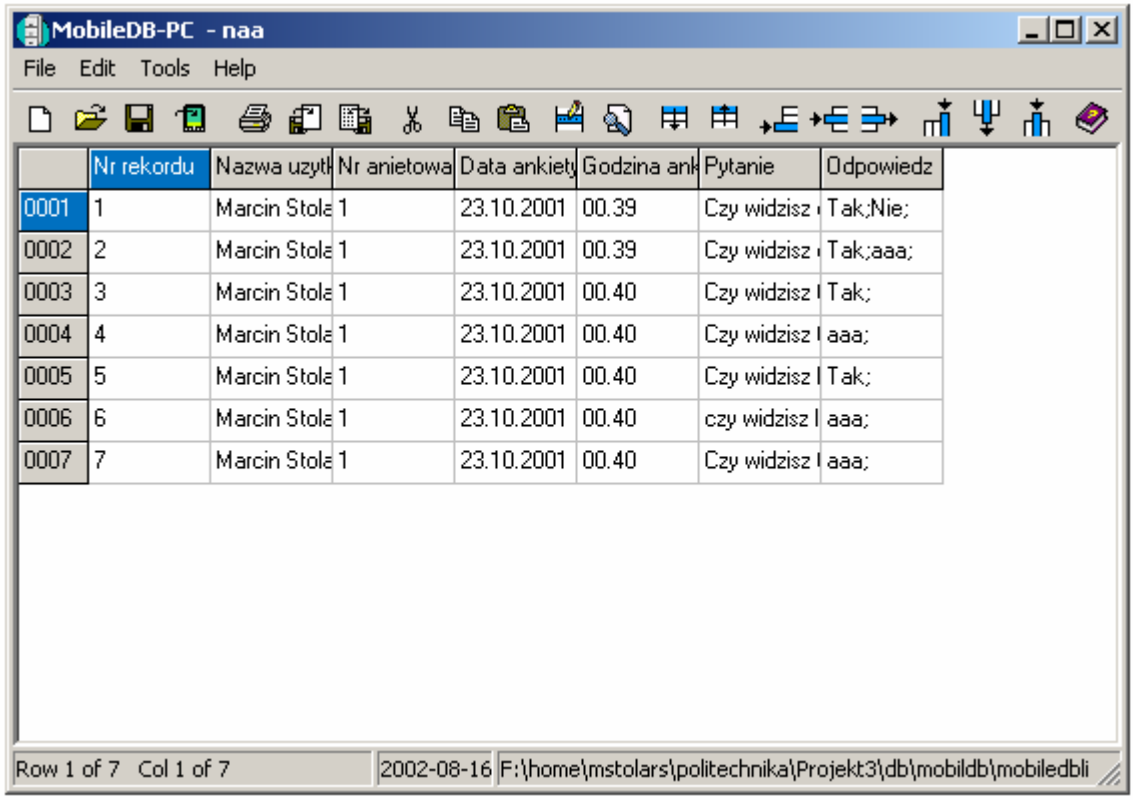
Odpowiedzi są zapisywane w podręcznej tablicy odpowiedzi zgodnie z poniżej przedstawionymi regułami.

Typ pytania	Zapisywana odpowiedz
checkboxy	Zapisywane są zaznaczone odpowiedzi jako ciągi znaków rozdzielone średnikami.
checkboxy + pole tekstowe	Zapisywane są zaznaczone odpowiedzi oraz pole txt (jeśli jest zaznaczone) jako ciągi znaków rozdzielone średnikami.
Trigery	Zapisywana jest tylko zaznaczona odpowiedź jako ciągi znaków.
trigery + pole tekstowe	Zapisywana jest tylko zaznaczona odpowiedź, bądź pole testowe (jeśli zostało wypełnione) jako ciągi znaków.
Lista	Zapisywana jest tylko zaznaczona odpowiedź jako ciągi znaków.
lista + pole tekstowe	Zapisywana jest tylko zaznaczona odpowiedź, bądź pole testowe (jeśli zostało wypełnione) jako ciągi znaków.
pole tekstowe	Zapisywane jest pole tekstowe jako ciągi znaków.

Gdy dojdziemy do końca ankiety [Rysunek 9] i naciśniemy przycisk [>>>] tablica podręczna jest kopiowana do bazy danych naa.pbd.

Rysunek 9. Koniec ankiety.

Jeśli zdecydujemy się żeby przenieść bazę danych z odpowiedziami na komputer stacjonarny możemy to zrobić wykorzystując program HotSync, bądź przesłać ją jako plik przy wykorzystaniu łącza podczerwieni (niestety palmptop nie pozwala na przesyłanie pojedynczych plików przez łącze kablowe). Na PC otwieramy plik naa.pbd [Rysunek 10].



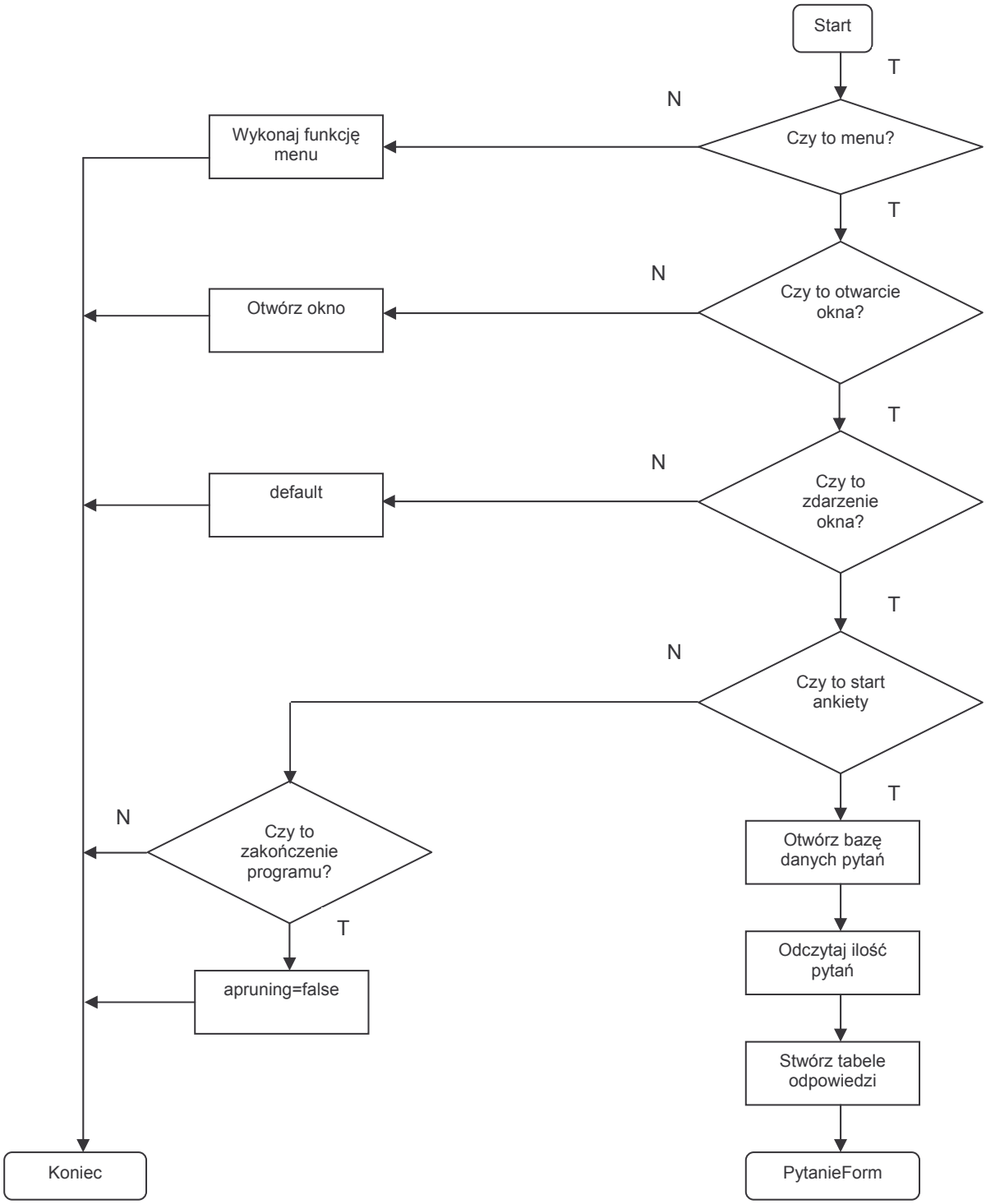
Rysunek 10. Plik odpowiedzi otwarty przez aplikacje MobilDB.

Jak widać tym razem jest to już tablica pięciokolumnowa zawierająca: Nr rekordu, Nazwę ankietera, Nr ankiety, Datę ankiety, Godzinę odpowiedzi, Treść pytania oraz Odpowiedz. Taką tabelę można wyeksportować do pliku txt.

```
{Column Data Types}     Text    Text    Text    Text    Text    Text    Text
Nr rekordu      Nazwa uzytkownika       Nr anietowanego Data ankiety    Godzina ankiety Pytanie
        Odpowiedz
1       Marcin Stolarski        1       23.10.2001      00.39   Czy     widzisz      checkboxy?
        Tak;Nie;
2       Marcin Stolarski        1       23.10.2001      00.39   Czy  widzisz  checkboxy  +  txt?
        Tak;aaa;
3       Marcin Stolarski        1       23.10.2001      00.40   Czy widzisz trigery?    Tak;
4       Marcin Stolarski        1       23.10.2001      00.40   Czy  widzisz  trigery  +  txt?
        aaa;
5       Marcin Stolarski        1       23.10.2001      00.40   Czy widzisz liste?      Tak;
6       Marcin Stolarski        1       23.10.2001      00.40   czy  widzisz  liste  +  txt?
        aaa;
7       Marcin Stolarski        1       23.10.2001      00.40   Czy widzisz txt?        aaa;
```

Podstawowe algorytmy

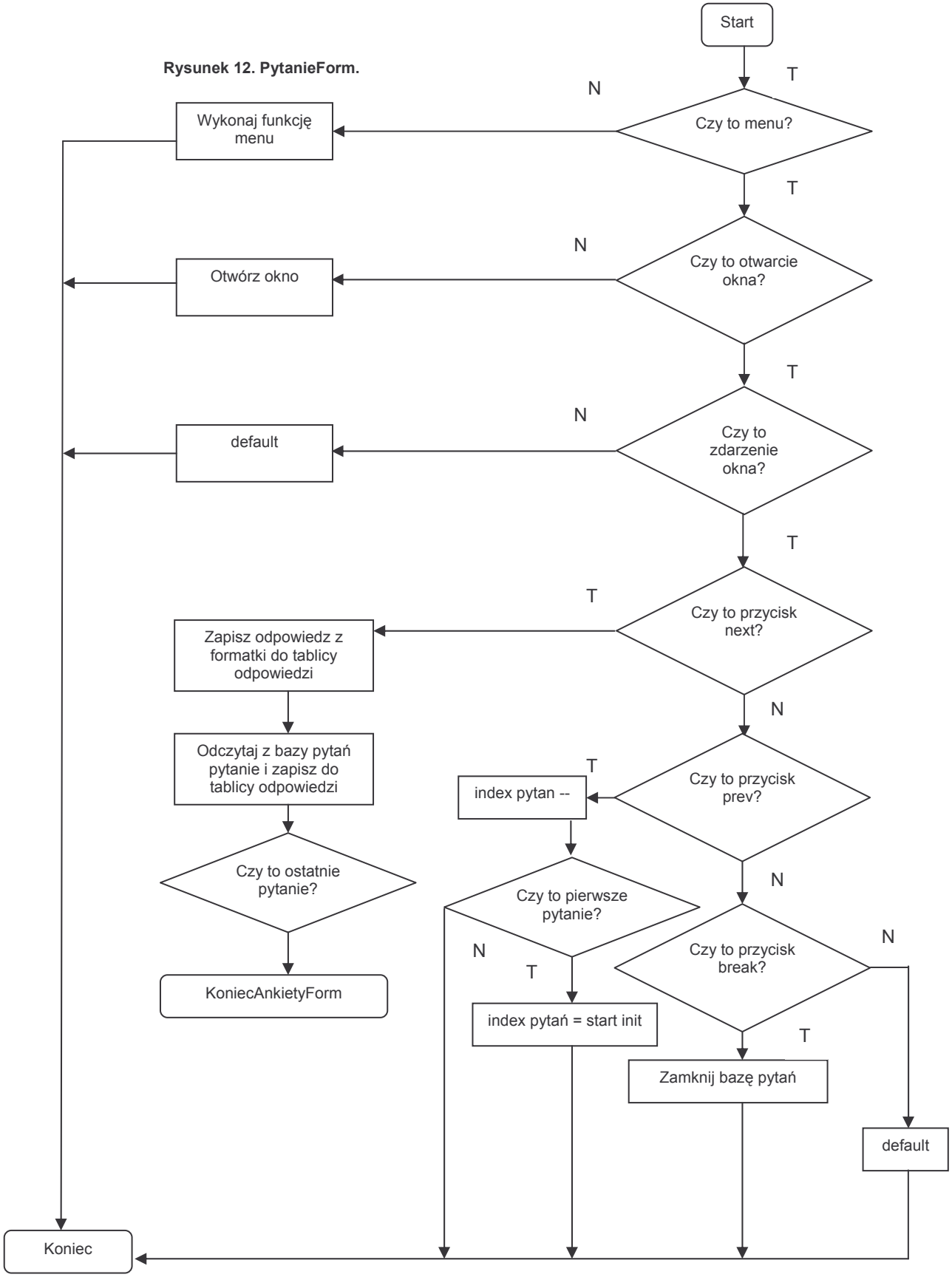
Kiedy mówimy o algorytmach w aplikacji PalmOS mówimy o pętlach zdarzeń. Podstawową pętlą jest MainFormHandleEvent [Rysunek 11]. Na początku sprawdzane są zdarzenia związane z budową formatki, potem z reakcjami użytkownika (naciśnięcia przycisków itp.).



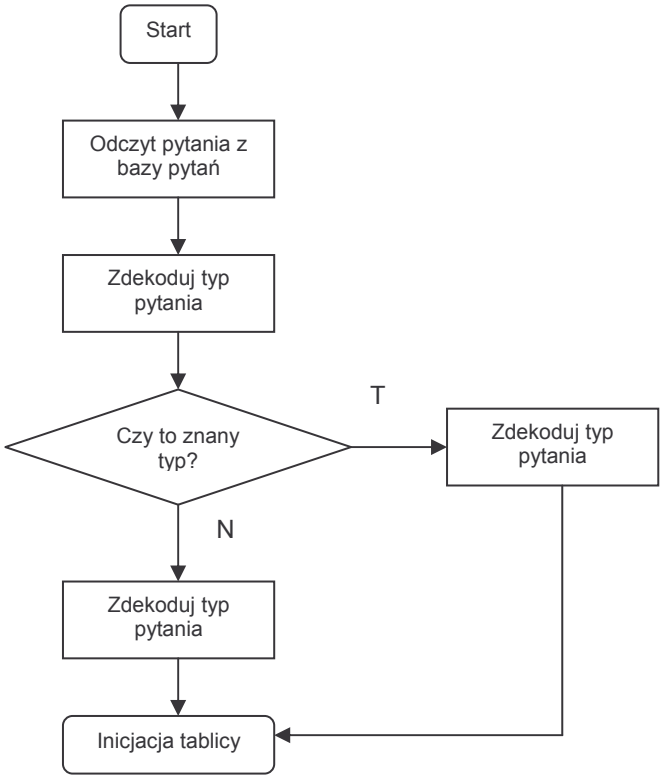
Rysunek 11. MainFormHandleEvent.

Jeśli w MainForm nacisnęliśmy przycisk rozpoczęcia ankiety następuje przejście do formatki PytanieFormHandleEvent [Rysunek 12] i tak jak poprzednio, najpierw obsługiwane są zdarzenia związane z otwarciem okna i budowaniem menu, a następnie z naciśnięciami przycisków formatki.

Rysunek 12. PytanieForm.

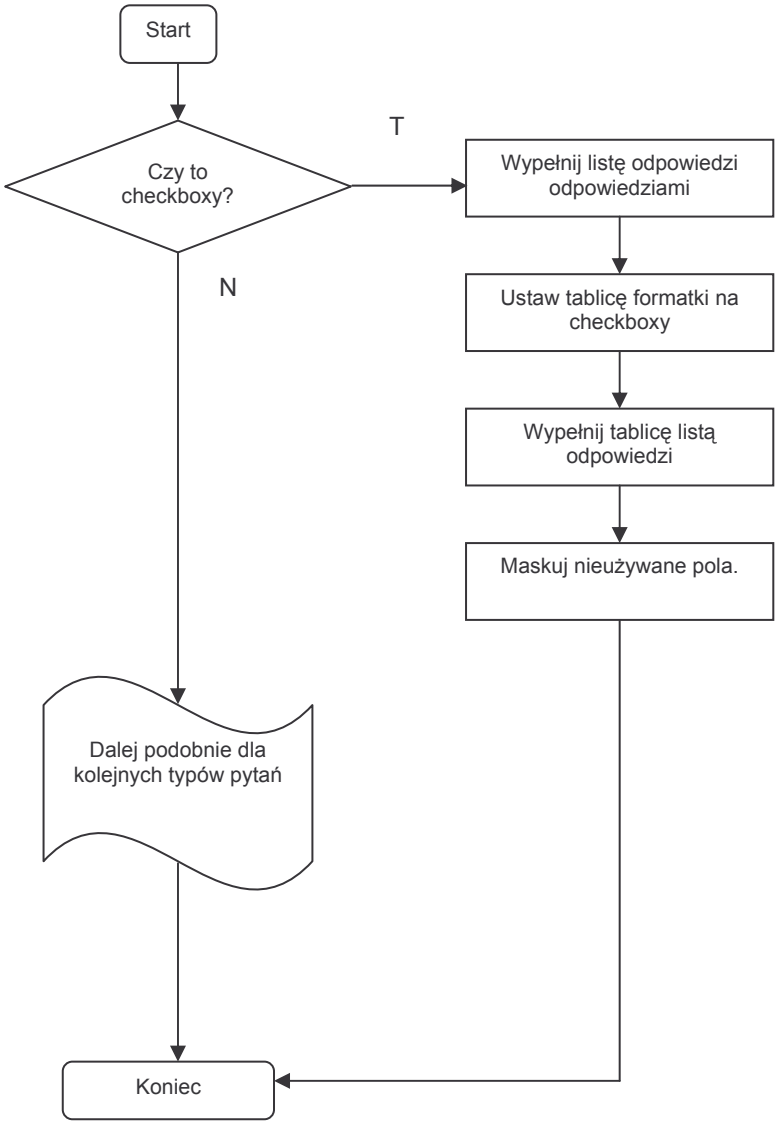


Podczas budowania formatki PytanieForm wywoływana jest funkcja UstawPyt [Rysunek 13], która doczytuje pytanie z bazy danych naq.pbd a następnie wyświetla je w formatce, po czym woła funkcję InicjacjaTablicy



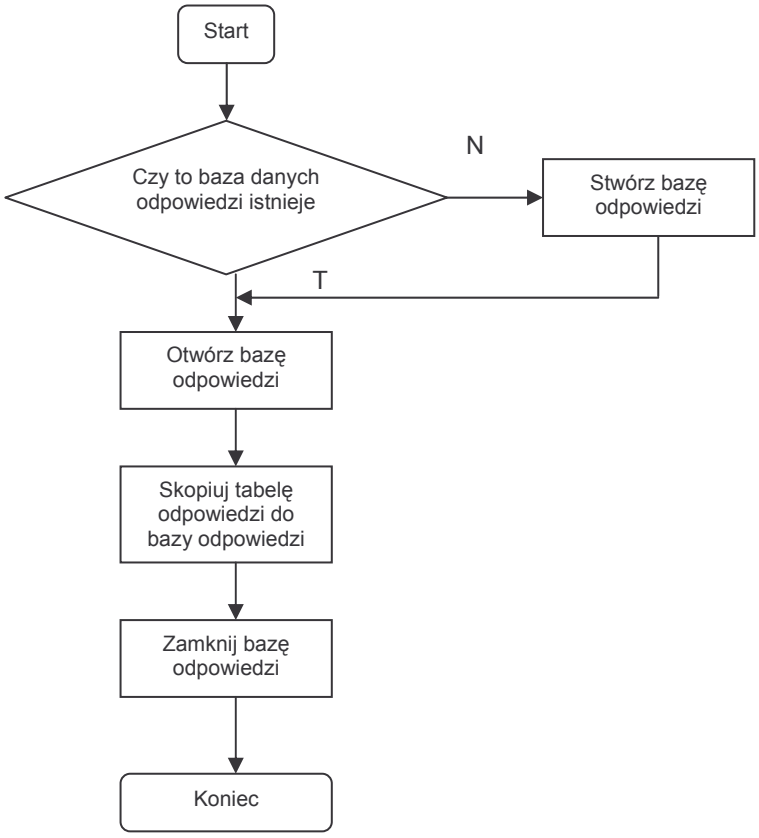
Rysunek 13. UstawPyt.

Funkcja InicjacjaTablicy [Rysunek 14] na podstawie przekazanego typu pytania wypełnia tablice odpowiedzi odpowiedziami i ustawia stosowne kontrolki (np. checkboxy).



Rysunek 14. InicjacjaTablicy.

Kiedy dochodzimy do końca ankiety i naciśniemy przycisk zapisz ankietę, wywoływana jest funkcja ZapiszAnkietę [Rysunek 15], która przepisuje pytania, odpowiedzi i inne pola z podręcznej tablicy odpowiedzi do bazy danych odpowiedzi naa.pbd. Gdyby taki plik nie istniał, jest tworzony nowy (w standardzie MobilDB).



Rysunek 15. ZapiszAnkietę.

Kod programu

Kod programu umieszczony jest w załączniku 1.

Podsumowanie

Celem tej pracy było napisanie aplikacji na palmptopa do przeprowadzania ulicznych ankiet. Cel ten został osiągnięty (aplikacja jest opisana w dokumencie), niemniej uważam, że nie jest to tak istotne jak zdobycie przeze mnie umiejętności pisania aplikacji na system PalmOS. W pracy tej można znaleźć przykład od czego zacząć, jakie wybrać oprogramowanie, gdzie szukać pomocy.

Generalnie na początku proponuję zaznajomienie się z informacjami zawartymi w dokumentacji na CD odnośnie komputerów PalmOS. Potem wybranie jednej z proponowanych książek, wybranie środowiska programistycznego, przejrzenie kodu, oraz opisu aplikacji Notatnik Ankietera. Funkcjonalność aplikacji wybrana jest specjalnie, jako że tego typu aplikacje są w tej chwili najbardziej poszukiwane na rynku (o czym autor miał się okazję przekonać osobiście). Można w niej znaleźć przykłady na pracę z plikami (bazami danych), wymianę informacji z innymi aplikacjami i systemami, tworzenie i edycję wyglądu formatki. Kolejnymi etapami powinno być wyposażenie aplikacji w narzędzia komunikacji sieciowej (połączenie z bazą danych online, przesyłanie danych przy wykorzystaniu telefonii komórkowej itp.) ale to już temat na kolejną pracę.

Podczas pisania programu okazało się, że największe trudności pojawiły się w miejscu, gdzie autor chciał sobie ułatwić, a mianowicie podczas zapisu i odczytu plików w formacie MobilDB. Okazało się, że firma zajmująca się rozprowadzaniem tego oprogramowania nie wspiera programistów i nie potrafi udzielić informacji odnośnie samego formatu plików. Gdybym nie znalazł w sieci kodu aplikacji wykorzystującej ten format, mogłoby się okazać konieczne napisanie własnych narzędzi na PC.

Podczas testów aplikacji okazało się, że komputery przenośne nadają się jako urządzenia do ankietowania ulicznego, ale trzeba sobie zdawać z ograniczeń, to znaczy małego ekranu palmptopa. Pytania muszą być krótkie, a proponowane odpowiedzi w formie co najwyżej paru wyrazów. Być może wprowadzenie e-booków (elektronicznych książek formatu A4) pozwoli na rozwiązanie tego typu problemów, ale to przyszłość najbliższych pięciu lat, że względu na trwające prace nad energooszczędnymi wyświetlaczami formatu A4 (technologie, e-ink, oled, ekrany polimerowe).

Do pracy tej jest dołączony CD-ROM, na którym można znaleźć wersje demonstracyjne środowisk programistycznych, dodatkowe materiały i załączniki, książki elektroniczne, kod źródłowy aplikacji i inne pomocne rzeczy, by samodzielnie móc rozpocząć swoją przygodę z programowaniem aplikacji na PalmOS.

Aby zobaczyć aplikacje Notatnik Ankietera w akcji należy:

- zainstalować emulator Palm (POSE)
- przygotować ankietę bądź wykorzystać gotową ankietę qaa.pbd
- uruchomić POSE i wgrać do niego (metodą przeciągnij i upuść) pliki na.prc, oraz naq.pbd (oraz ewentualnie aplikację MobilDB)
- uruchomić dowolną aplikację w POSE (np. kalkulator) i wyjść z niej (nacisnąć domek) aby odświeżyć pulpit (powinna pojawić się ikona Notatnik Ankietera)
- uruchomić aplikacje NA
- przejść przez ankietę
- zamknąć aplikację (domek)
- wyeksportować plik naa.pbd (prawy przycisk myszy) i otworzyć za pomocą MobilDB-PC, bądź obejrzeć wyniki ankiety na samym emulatorze za pomocą aplikacji MobilDB

Załączniki

Załącznik 1: Kod programu

```
//      Header generated by Constructor for Palm OS® 1.6
//
//      Generated at 23:31:12  on 18 lipiec 2002
//
//      Generated for file: F:\home\mstolars\politechnika\Projekt3\palm\NA\na\Rsc\Starter.rsrc
//
//      THIS IS AN AUTOMATICALLY GENERATED HEADER FILE FROM CONSTRUCTOR FOR PALM OS®;
//      - DO NOT EDIT - CHANGES MADE TO THIS FILE WILL BE LOST
//
//      Palm App Name:                  "Notatnik Ankietera"
//
//      Palm App Version:               "1.0"


//      Resource: tFRM 1000
#define MainForm                                1000 //(Left Origin = 0, Top Origin = 0,
Width = 160, Height = 160, Usable = 1, Modal = 0, Save Behind = 0, Help ID = 0, Menu Bar ID =
1000, Default Button ID = 1001)
#define MainRozpocznijankiete1001Button         1001 //(Left Origin = 79, Top Origin = 147,
Width = 80, Height = 12, Usable = 1, Anchor Left = 1, Frame = 1, Non-bold Frame = 1, Font =
Standard)
#define MainZakoncz1002Button                   1002 //(Left Origin = 1, Top Origin = 147,
Width = 40, Height = 12, Usable = 1, Anchor Left = 1, Frame = 1, Non-bold Frame = 1, Font =
Standard)
#define MainMobilisRex1101Label                 1003 //(Left Origin = 49, Top Origin = 102,
Usable = 1, Font = Bold 12)
#define MainTitleLabel                          1004 //(Left Origin = 27, Top Origin = 25,
Usable = 1, Font = Bold 12)
#define MainWww1106Label                        1007 //(Left Origin = 26, Top Origin = 117,
Usable = 1, Font = Standard)
#define MainCopyright1107Label                  1008 //(Left Origin = 19, Top Origin = 86,
Usable = 1, Font = Bold)

//      Resource: tFRM 1100
#define AboutForm                               1100 //(Left Origin = 2, Top Origin = 2,
Width = 156, Height = 156, Usable = 1, Modal = 1, Save Behind = 1, Help ID = 0, Menu Bar ID =
0, Default Button ID = 0)
#define AboutOKButton                           1105 //(Left Origin = 55, Top Origin = 133,
Width = 40, Height = 12, Usable = 1, Anchor Left = 1, Frame = 1, Non-bold Frame = 1, Font =
Standard)
#define AboutMobilisRex1101Label                1101 //(Left Origin = 48, Top Origin = 102,
Usable = 1, Font = Bold 12)
#define AboutTitleLabel                         1102 //(Left Origin = 26, Top Origin = 25,
Usable = 1, Font = Bold 12)
#define AboutText1Label                         1103 //(Left Origin = 11, Top Origin = 61,
Usable = 1, Font = Standard)
#define AboutText2Label                         1104 //(Left Origin = 47, Top Origin = 42,
Usable = 1, Font = Bold)
```

```
#define AboutWww1106Label                       1106 //(Left Origin = 25, Top Origin = 117,
Usable = 1, Font = Standard)
#define AboutCopyright1107Label                 1107 //(Left Origin = 18, Top Origin = 86,
Usable = 1, Font = Bold)

//      Resource: tFRM 1200
#define PytanieForm                             1200 //(Left Origin = 0, Top Origin = 0,
Width = 160, Height = 160, Usable = 1, Modal = 0, Save Behind = 0, Help ID = 0, Menu Bar ID =
1000, Default Button ID = 1204)
#define PytanieNastepny1204Button               1204 //(Left Origin = 123, Top Origin = 147,
Width = 36, Height = 12, Usable = 1, Anchor Left = 1, Frame = 1, Non-bold Frame = 1, Font =
Bold)
#define PytaniePoprzedni1205Button              1205 //(Left Origin = 82, Top Origin = 147,
Width = 36, Height = 12, Usable = 1, Anchor Left = 1, Frame = 1, Non-bold Frame = 1, Font =
Bold)
#define PytanieZakoncz1206Button                1206 //(Left Origin = 1, Top Origin = 147,
Width = 36, Height = 12, Usable = 1, Anchor Left = 1, Frame = 1, Non-bold Frame = 1, Font =
Standard)
#define PytanieNrPytania1202Field               1202 //(Left Origin = 50, Top Origin = 15,
Width = 36, Height = 12, Usable = 1, Editable = 1, Underline = 0, Single Line = 0, Dynamic
Size = 0, Left Justified = 1, Max Characters = 3, Font = Standard, Auto Shift = 0, Has Scroll
Bar = 0, Numeric = 0)
#define PytaniePytanie1203Field                 1203 //(Left Origin = 0, Top Origin = 27,
Width = 160, Height = 48, Usable = 1, Editable = 1, Underline = 0, Single Line = 0, Dynamic
Size = 0, Left Justified = 1, Max Characters = 80, Font = Standard, Auto Shift = 0, Has Scroll
Bar = 1, Numeric = 0)
#define PytanieNrPytania1201Label               1201 //(Left Origin = 0, Top Origin = 15,
Usable = 1, Font = Standard)
#define PytanieLista1208List                    1208 //(Left Origin = 20, Top Origin = 15,
Width = 140, Usable = 0, Font = Standard, Visible Items = 0)
#define PytanieTablica1207Table                 1207 //(Left Origin = 0, Top Origin = 75,
Width = 160, Height = 70, Editable = 1, Rows = 5)

//      Resource: tFRM 1300
#define KoniecAnkietyForm                       1300 //(Left Origin = 0, Top Origin = 0,
Width = 160, Height = 160, Usable = 1, Modal = 0, Save Behind = 0, Help ID = 0, Menu Bar ID =
1000, Default Button ID = 0)
#define KoniecAnkietyZakoncz1206Button          1301 //(Left Origin = 1, Top Origin = 147,
Width = 36, Height = 12, Usable = 1, Anchor Left = 1, Frame = 1, Non-bold Frame = 1, Font =
Standard)
#define KoniecAnkietyPoprzedni1205Button        1302 //(Left Origin = 82, Top Origin = 147,
Width = 36, Height = 12, Usable = 1, Anchor Left = 1, Frame = 1, Non-bold Frame = 1, Font =
Bold)
#define KoniecAnkietyNastepny1204Button         1303 //(Left Origin = 123, Top Origin = 147,
Width = 36, Height = 12, Usable = 1, Anchor Left = 1, Frame = 1, Non-bold Frame = 1, Font =
Bold)
#define KoniecAnkietyUnnamed1304Label           1304 //(Left Origin = 46, Top Origin = 61,
Usable = 1, Font = Bold 12)


//      Resource: Talt 1001
#define RomIncompatibleAlert                    1001
#define RomIncompatibleOK                       0

//      Resource: Talt 1000
```

```
#define InfoAlert                               1000
#define InfoOK                                  0


//      Resource: MBAR 1000
#define MainFormMenuBar                         1000


//      Resource: MENU 1000
#define MainOptionsMenu                         1000
#define MainOptionsOprogramie                   1000


//      Resource: PICT 1
#define BitmapID1Bitmap                         1

//      Resource: PICT 2
#define BitmapID2Bitmap                         2

//      Resource: PICT 3
#define BitmapID3Bitmap                         3

//      Resource: PICT 4
#define BitmapID4Bitmap                         4


//      Resource: taif 1000
#define LargeAppIconAppIconFamily               1000

//      Resource: taif 1001
#define SmallAppIconAppIconFamily               1001
```

```
/***********************************************************************
 * Notatnik Ankietera
 * Copyright (c) 2002 by Marcin Stolarski
 *
 * Mobilis Rex
 * http://mobilisrex.prv.pl
 * All rights reserved.
 *
 * File: Starter.cpp
 *
 ***********************************************************************/

#include <PalmOS.h>
#include "StarterRsc.h"
#include <DLServer.h>
#include <ExgMgr.h>

/***********************************************************************
 *
 *   Internal Constants
 *
 ***********************************************************************/
#define appFileCreator                   'MRna'
#define appVersionNum             0x01
#define appPrefID                 0x00
#define appPrefVersionNum         0x01


#define CATEGORY_FIELD_LABELS          1
#define CATEGORY_DATA_RECORDS          2
#define CATEGORY_DATA_RECORDS_FILTERED 3
#define CATEGORY_PREFERENCES           4
#define CATEGORY_DATA_TYPES            5
#define CATEGORY_FIELD_LENGTHS         6

#define MAX_FIELDS 20

typedef struct date_struct {
  UInt16 year;
  UInt8 month;
  UInt8 day;
} Date;

typedef struct time_struct {
  UInt8 hour;
  UInt8 min;
} Time;


typedef union {
  double d;
  char *s;
  UInt16 uh;
  UInt32 ui;
  Int32   i;
```

```
  Date ymd;
  Time t;
} FieldValueUnion;

typedef struct calculated_value_struct {

  UInt8 info;
  FieldValueUnion value;

} CalculatedValue;

/* Function pointers to retrieve specific fields of a record. */
/* !! all getter functions used by Global Search must be in the main section*/
typedef struct data_source_getter
{
    char *  (*GetString) (void * data, UInt16 fieldNum);
    Int32   (*GetInteger)(void * data, UInt16 fieldNum);
    Boolean (*GetBoolean)(void * data, UInt16 fieldNum);
    void    (*GetDate)(void * data, UInt16 fieldNum,
                       UInt16* year, UInt8* mon, UInt8* day);
    void    (*GetTime)(void * data, UInt16 fieldNum,
                       UInt8* hour, UInt8* minute);
    char *  (*GetNote)(void * data, UInt16 fieldNum);
    char *  (*GetNoteTitle)(void * data, UInt16 fieldNum);
    UInt8    (*GetListItem)(void * data, UInt16 fieldNum);
    char *  (*GetLink)(void * data, UInt16 fieldNum, UInt32* uniqID);
    char *  (*GetLinked)(void * data, UInt16 fieldNum);
    double (*GetFloat)(void * data, UInt16 fieldNum);
    void  (*GetCalculated) (void * data, UInt16 fieldNum, CalculatedValue *cv);
} DataSourceGetter;

typedef struct {
    char  field[40];   /* the filter text the user entered */
    Int8  fieldNo;     /* number of the field, -1 means any field */
    UInt8 flags;       /* reserved */
}  FilterCriterion;

typedef struct {
    Int8 fieldNo;         /* field # to sort on, -1 means unused criterion */
    Boolean descending;   /* direction to sort */
    UInt8 type;           /* lowest bit: 1 means numeric, 0 means lexical */
} SortCriterion;

/* structure which comprises the application information block */
typedef struct {
    UInt16  renamedCategories; /* should be 0x0000 */
    char    categoryLabels[dmRecNumCategories][dmCategoryLength];
    UInt8   categoryUniqIDs[dmRecNumCategories]; /* 0 .. 15 */
    UInt8   lastUniqID;     /* 15 */

    UInt16  version;         /* current header version */
    UInt32  lock;            /* hash of password */
    Boolean dontSearch;   /* true, if DB should be invisible to find */
    Boolean editOnSelect; /* true, if record should be edited by default */
    UInt8   reserved[3];     /* for later use */
```

```
    FilterCriterion filter[3]; /* what the user typed in filters */
    SortCriterion sort[3];     /* what the user typed in sort */
} MobileDBInfoType;

static Char *   NameOdp[] = {"Nr rekordu", "Nazwa uzytkownika", "Nr anietowanego", "Data
ankiety", "Godzina ankiety", "Pytanie" ,"Odpowiedz"};

//baza pytan
#define libDBName               "naq"
#define libDBType               'Mdb1'
#define libCreatorID    'Mdb1'
#define DBqIndexStart   5
//Index bazy
UInt16 DBqindex=DBqIndexStart;
UInt16 DBqnext=0;
UInt16 DBqend=0;

DmOpenRef gLibDB;

//baza odpowiedzi
#define libDBNamea              "naa"
#define libDBTypea              'Mdb1'
#define libCreatorIDa   'Mdb1'
#define DBqIndexStarta  5
//Index bazy odpowiedzi
UInt16 DBqindexa=DBqIndexStarta;
UInt16 DBqnexta=0;
UInt16 DBqenda=0;

//typ aktywnego pytania
UInt16 aqtyp;

//ilosc odpowiedzi aktywnego pytania
UInt16 aqni;

DmOpenRef gLibDBa;

//Struktura rekordu bazy
        typedef struct {
                char txt[1000];
        } MyRecordType;

//Struktura rekordu odpowiedzi
typedef struct {
        char Lp[10];
        char Owner[50];
        char NrAnk[10];
        char Data[20];
        char Time[20];
        char Pyt[100];
        char Odp[500];
        } MyOdpType;

// Define the minimum OS version we support (2.0 for now).
```

```
#define ourMinVersion   sysMakeROMVersion(2,0,0,sysROMStageRelease,0)

// Table constants
#define numTextColumns   1
#define numTableColumns  2
#define numTableRows     5

//list root
char** blptr;
UInt16 blptrsize=0;

//lista odpowiedzi ankiety
MyOdpType** loaptr;
UInt16 loaptrsize=0;

char OdpStr[100];

/***********************************************************************
 *
 *   Entry Points
 *
 ***********************************************************************/

/***********************************************************************
 *
 *   Internal Structures
 *
 ***********************************************************************/
typedef struct
        {
        UInt8 replaceme;
        } StarterPreferenceType;

typedef struct
        {
        UInt8 replaceme;
        } StarterAppInfoType;

typedef StarterAppInfoType* StarterAppInfoPtr;


/***********************************************************************
 *
 *   Global variables
 *
 ***********************************************************************/
//static Boolean HideSecretRecords;

Boolean apruning=true; //do zatrzymywania aplikacji
Int32 nrpyt;

//static Char *   ListaOdp[] = {"Odpowiedz 1", "Odpowiedz 2", "Odpowiedz 3", "Odpowiedz 4",
"Odpowiedz 5", "Inne"};
Char** ListaOdp;
```

```
//static Char *   gLabels[] = {"Odpowiedz 1", "Odpowiedz 2", "Odpowiedz 3", "Odpowiedz 4",
"Odpowiedz 5", "Inne"};

MemHandle  gTextHandles[numTextColumns][numTableRows];

Boolean    gHideRows = true;
Boolean    gHideColumns = true;



/***********************************************************************
 *
 *   Internal Functions
 *
 ***********************************************************************/
static char * StrAdd(char *, char *);

/***********************************************************************
 *
 * FUNCTION:    RomVersionCompatible
 *
 * DESCRIPTION: This routine checks that a ROM version is meet your
 *              minimum requirement.
 *
 * PARAMETERS:  requiredVersion - minimum rom version required
 *                                (see sysFtrNumROMVersion in SystemMgr.h
 *                                for format)
 *              launchFlags     - flags that indicate if the application
 *                                UI is initialized.
 *
 * RETURNED:    error code or zero if rom is compatible
 *
 * REVISION HISTORY:
 *
 *
 ***********************************************************************/
static Err RomVersionCompatible(UInt32 requiredVersion, UInt16 launchFlags)
{
        UInt32 romVersion;

        // See if we're on in minimum required version of the ROM or later.
        FtrGet(sysFtrCreator, sysFtrNumROMVersion, &romVersion);
        if (romVersion < requiredVersion)
                {
                if ((launchFlags & (sysAppLaunchFlagNewGlobals | sysAppLaunchFlagUIApp)) ==
                        (sysAppLaunchFlagNewGlobals | sysAppLaunchFlagUIApp))
                        {
                        FrmAlert (RomIncompatibleAlert);

                        // Palm OS 1.0 will continuously relaunch this app unless we switch to
                        // another safe one.
                        if (romVersion < ourMinVersion)
                                {
                                AppLaunchWithCommand(sysFileCDefaultApp,
sysAppLaunchCmdNormalLaunch, NULL);
```

```
                                }
                        }

                return sysErrRomIncompatible;
                }

        return errNone;
}

/***********************************************************************
 *
 * FUNCTION:    GetTextColumn
 *
 * DESCRIPTION: Returns the text column corresponding to a specific
 *              table column.
 *
 * PARAMETERS:  formId - id of the form to display
 *
 * RETURNED:    void *
 *
 * REVISION HISTORY:
 *
 *
 ***********************************************************************/
static Int16 GetTextColumn(Int16 column)
{
        Int16  result=column;


        switch (column) {
                case 0:
                        result = 0;
                        break;

                case 1:
                        result = 0;
                        break;

                default:
                        result = 0;
                        break;
        }

        return result;
}

/***********************************************************************
 *
 * FUNCTION:    GetObjectPtr
 *
 * DESCRIPTION: This routine returns a pointer to an object in the current
 *              form.
 *
 * PARAMETERS:  formId - id of the form to display
 *
```

```
 * RETURNED:    void *
 *
 * REVISION HISTORY:
 *
 *
 ***********************************************************************/
static void * GetObjectPtr(UInt16 objectID)
{
        FormPtr frmP;

        frmP = FrmGetActiveForm();
        return FrmGetObjectPtr(frmP, FrmGetObjectIndex(frmP, objectID));
}

/***********************************************************************
 *
 * FUNCTION:    LoadTextTableItem
 *
 * DESCRIPTION: Callback routine to load the text of a table item.
 *
 * PARAMETERS:  table - pointer to the table object
 *              row - row in the table to load
 *              column - column in the table to load
 *              editable - true if a text field in the table is currently
 *                         being edited, false otherwise
 *              dataH - unlocked handle of a block containing a null-
 *                      terminated text string
 *              dataOffset - offset from the start of the block to the
 *                           start of the actual string
 *              dataSize - allocated size of the text string, NOT the
 *                         string length
 *              field - pointer to the text field in this table cell
 *
 *
 * RETURNED:    0 upon success, or an error code if unsuccessful
 *
 * REVISION HISTORY:
 *
 *
 ***********************************************************************/
static Err LoadTextTableItem(void *table, Int16 row, Int16 column,
                                                    Boolean editable, MemHandle *dataH,
                                                    Int16 *dataOffset, Int16 *dataSize,
                                                    FieldPtr field)
{
#ifdef __GNUC__
        CALLBACK_PROLOGUE;
#endif
                        /*
                        int i=column, j=row;
                        char al[50], ab[10];
                        StrCopy(al, "i=");
                        StrAdd(al, StrIToA(ab, i));
                        StrAdd(al, " j=");
                        StrAdd(al, StrIToA(ab, j));
```

```
                        FrmCustomAlert(InfoAlert, al, NULL, NULL);
                        */
        *dataH = gTextHandles[GetTextColumn(column)][row];

        char * str;
        str = (char*)MemHandleLock(gTextHandles[GetTextColumn(column)][row]);
        if (row!=aqni&&aqtyp>0&&aqtyp<5) StrCopy(str, ListaOdp[row]);
                else *str = '\0';

        MemHandleUnlock(gTextHandles[GetTextColumn(column)][row]);

        *dataOffset = 0;
        *dataSize = MemHandleSize(*dataH);

#ifdef __GNUC__
        CALLBACK_EPILOGUE;
#endif

        return 0;

}


static Boolean SaveTextTableItem(void *table, Int16 row, Int16 column)
{
        Boolean result = false;
        FieldType *field;
        MemHandle textH;
        Char *str;
        //Int16 i;
        field = TblGetCurrentField((TableType*)table);
        // If the field has been changed, uppercase its text.
        if (field && FldDirty(field)) {
                textH = gTextHandles[GetTextColumn(column)][row];
                str = (char*)   MemHandleLock(textH);
/*              //Zamien na duze litery
                for (i = 0; str[i] != '\0'; i++) {
                        if (str[i] >= 'a' && str[i] <= 'z') {
                        str[i] -= 'a' - 'A';
                        }
                }
*/
                StrCopy(OdpStr, str);
                MemHandleUnlock(textH);
                TblMarkRowInvalid((TableType*)table, row);
                result = true;
        }

        return result;
}


/***********************************************************************
 *
 * FUNCTION:    MainFormInit
```

```
 *
 * DESCRIPTION: This routine initializes the MainForm form.
 *
 * PARAMETERS:  frm - pointer to the MainForm form.
 *
 * RETURNED:    nothing
 *
 * REVISION HISTORY:
 *
 *
 ***********************************************************************/
static void MainFormInit(FormPtr /*frmP*/)
{
}


/***********************************************************************
 *
 * FUNCTION:    MainFormDoCommand
 *
 * DESCRIPTION: This routine performs the menu command specified.
 *
 * PARAMETERS:  command  - menu item id
 *
 * RETURNED:    nothing
 *
 * REVISION HISTORY:
 *
 *
 ***********************************************************************/
static Boolean MainFormDoCommand(UInt16 command)
{
        Boolean handled = false;
   FormPtr frmP;

        switch (command)
                {
                case MainOptionsOprogramie:
                        MenuEraseStatus(0);                                     //   Clear   the
menu status from the display.
                        frmP = FrmInitForm (AboutForm);
                        FrmDoDialog (frmP);                                     //  Display  the
About Box.
                        FrmDeleteForm (frmP);
                        handled = true;
                        break;

                }

        return handled;
}

static Boolean PytanieFormDoCommand(UInt16 command)
{
        Boolean handled = false;
```

```
   FormPtr frmP;

        switch (command)
                {
                case MainOptionsOprogramie:
                        MenuEraseStatus(0);                                     //   Clear   the
menu status from the display.
                        frmP = FrmInitForm (AboutForm);
                        FrmDoDialog (frmP);                                     //  Display  the
About Box.
                        FrmDeleteForm (frmP);
                        handled = true;
                        break;

                }

        return handled;
}

/***********************************************************************
 *
 *   Funkcje programu
 *
 ***********************************************************************/


/*
//Wysylanie bazy danych przez IR
Err WriteDatabase (const void *buffer, UInt32 *bytes, void *exgSocket) {
        Err error;
        ExgSend( (ExgSocketType *) exgSocket, buffer, bytes, &error);
        return error;
}

Err SendDatabase (Char *dbName) {
        ExgSocketType exgSocket;
        Err error;
        Char name[36]; // max length for a database name (32),
        // plus a period and three-letter file
        // extension
        // Initialize the socket structure.
        MemSet(&exgSocket, sizeof(exgSocket), 0);
        StrCopy(name, dbName);
        //StrCat(name, ".prc");
        exgSocket.description = dbName;
        exgSocket.name = name;
        // Make a connection to send the database.
        error = ExgPut(&exgSocket);
        if (! error) {
                error = ExgDBWrite(WriteDatabase, &exgSocket, dbName, NULL, 0);
                error = ExgDisconnect(&exgSocket, error);
                }
        return error;
        }
*/
```

```
//dodawanie stringow. Uwaga, dest musi byc odpowiednio dlugo zarezerwowany w pamieci !!!
static char* StrAdd(char * dest, char * source) {
        //char buf[200];
        UInt16 x,n=StrLen(dest);


        for(x=0;x<StrLen(source);x++) dest[x+n]=source[x];
        dest[x+n]=0;

        return dest;
}

static void InicjacjaTablicy(FormPtr form, Int16 typtablicy=0, UInt16 qni=0)
{
        TablePtr  table;
        Int16     numRows;
        Int16     i;
        ListType* l;

        MemHandle h;



        table = (TablePtr) FrmGetObjectPtr(form, FrmGetObjectIndex(form, 1207));
        TblEraseTable(table);
        numRows = TblGetNumberOfRows(table);

        switch(typtablicy) {
                case 1: //checkbox
                        for (i = 0; i < qni; i++) {
                                //odczyt odpowiedzi z bazy
                                h=DmQueryRecord(gLibDB, DBqindex+2+i);
                                if (h) {
                                        MyRecordType *p=(MyRecordType *)MemHandleLock(h);
                                        //odczyt rekordu
                                        StrCopy(ListaOdp[i], &p->txt[8]);
                                        //FrmCustomAlert(InfoAlert, &p->txt[8], NULL, NULL);
                                        MemPtrUnlock(p);
                                        }

                                TblSetItemStyle(table, i, 0, checkboxTableItem);
                                TblSetItemInt(table, i, 0, 0);

                                TblSetItemStyle(table, i, 1, labelTableItem);
                                TblSetItemPtr(table, i, 1, ListaOdp[i]);

                                TblSetRowStaticHeight(table, i, true);
                                TblSetRowUsable(table, i, true);
                                }
                                for (i++; i < numRows; i++) {
                                TblSetRowUsable(table, i, false);
                                }

                                for (i = 0; i <= numTableColumns; i++) {
```

```
                        TblSetColumnUsable(table, i, true);
                        TblSetColumnSpacing(table, i, 0);
                }

                break;

case 2: //checkbox + txt
        for (i = 0; i < qni; i++) {
                //odczyt odpowiedzi z bazy
                h=DmQueryRecord(gLibDB, DBqindex+2+i);
                if (h) {
                        MyRecordType *p=(MyRecordType *)MemHandleLock(h);
                        //odczyt rekordu
                        StrCopy(ListaOdp[i], &p->txt[8]);
                        //FrmCustomAlert(InfoAlert, &p->txt[8], NULL, NULL);
                        MemPtrUnlock(p);
                        }

                TblSetItemStyle(table, i, 0, checkboxTableItem);
                TblSetItemInt(table, i, 0, 0);

                //TblSetItemStyle(table, i, 1, textTableItem);
                TblSetItemStyle(table, i, 1, labelTableItem);
                TblSetItemPtr(table, i, 1, ListaOdp[i]);

                TblSetRowStaticHeight(table, i, true);
                TblSetRowUsable(table, i, true);
                }
                TblSetItemStyle(table, i, 0, labelTableItem);
                TblSetItemPtr(table, i, 0, "Inne");
                TblSetItemStyle(table, i, 1, textTableItem);
                TblSetRowUsable(table, i, true);

                for (i++; i < numRows; i++) {
                TblSetRowUsable(table, i, false);
                }

                for (i = 0; i <= numTableColumns; i++) {
                        TblSetColumnUsable(table, i, true);
                        TblSetColumnSpacing(table, i, 0);
                }

                break;

case 3: //triger
        for (i = 0; i < qni; i++) {
                //odczyt odpowiedzi z bazy
                h=DmQueryRecord(gLibDB, DBqindex+2+i);
                if (h) {
                        MyRecordType *p=(MyRecordType *)MemHandleLock(h);
                        //odczyt rekordu
                        StrCopy(ListaOdp[i], &p->txt[8]);
                        //FrmCustomAlert(InfoAlert, &p->txt[8], NULL, NULL);
                        MemPtrUnlock(p);
                        }
```

```
                        TblSetItemStyle(table, i, 0, checkboxTableItem);
                        TblSetItemInt(table, i, 0, 0);

                        TblSetItemStyle(table, i, 1, labelTableItem);
                        TblSetItemPtr(table, i, 1, ListaOdp[i]);

                        TblSetRowStaticHeight(table, i, true);
                        TblSetRowUsable(table, i, true);
                        }
                        for (i++; i < numRows; i++) {
                        TblSetRowUsable(table, i, false);
                        }

                        for (i = 0; i <= numTableColumns; i++) {
                                TblSetColumnUsable(table, i, true);
                                TblSetColumnSpacing(table, i, 0);
                        }

                        break;

case 4: //triger + txt
        for (i = 0; i < qni; i++) {
                        //odczyt odpowiedzi z bazy
                        h=DmQueryRecord(gLibDB, DBqindex+2+i);
                        if (h) {
                                MyRecordType *p=(MyRecordType *)MemHandleLock(h);
                                //odczyt rekordu
                                StrCopy(ListaOdp[i], &p->txt[8]);
                                //FrmCustomAlert(InfoAlert, &p->txt[8], NULL, NULL);
                                MemPtrUnlock(p);
                                }

                        TblSetItemStyle(table, i, 0, checkboxTableItem);
                        TblSetItemInt(table, i, 0, 0);

                        TblSetItemStyle(table, i, 1, labelTableItem);
                        TblSetItemPtr(table, i, 1, ListaOdp[i]);

                        TblSetRowStaticHeight(table, i, true);
                        TblSetRowUsable(table, i, true);
                        }
                        TblSetItemStyle(table, i, 0, labelTableItem);
                        TblSetItemPtr(table, i, 0, "Inne");
                        TblSetItemStyle(table, i, 1, textTableItem);
                        TblSetRowUsable(table, i, true);
                        for (i++; i < numRows; i++) {
                        TblSetRowUsable(table, i, false);
                        }

                        for (i = 0; i <= numTableColumns; i++) {
                                TblSetColumnUsable(table, i, true);
                                TblSetColumnSpacing(table, i, 0);
                        }
```

```
                                break;



                        case 5: //list


                                if (blptr) {
                                        for (i=0; i<blptrsize; i++) {
                                                if(blptr[i]) MemPtrFree(blptr[i]);
                                                }
                                        MemPtrFree(blptr);
                                }

                                blptr=(char **)MemPtrNew(qni*sizeof(char *));
                                blptrsize=qni;

                                //int i;
                                //ListType* l;

                                l  =  (ListType*)FrmGetObjectPtr(form,  FrmGetObjectIndex(form,
1208));
                                /* ... */
                                //teraz wypelnienie tablicy wskaznikow (blptr) wartosciami
                                for (i=0; i<qni; i++) { //ilePozycji
                                //odczyt odpowiedzi z bazy
                                h=DmQueryRecord(gLibDB, DBqindex+2+i);
                                if (h) {
                                        MyRecordType *p=(MyRecordType *)MemHandleLock(h);
                                        //odczyt rekordu
                                        blptr[i]=(char *)MemPtrNew(30);

                                        StrCopy(blptr[i], &p->txt[8]);
                                        //FrmCustomAlert(InfoAlert, &p->txt[8], NULL, NULL);
                                        MemPtrUnlock(p);
                                        }
                                }

                                LstSetSelection(l, 0);
                                LstSetListChoices(l, blptr, qni);
                                LstSetHeight(l, qni);


                                TblSetItemStyle(table, 0, 0, labelTableItem);
                                TblSetItemPtr(table, 0, 0, "->");

                                TblSetItemStyle(table, 0, 1, popupTriggerTableItem);
                                TblSetItemInt(table, 0, 1, 0);
                                TblSetItemPtr(table, 0, 1, l);

                                for (i = 0; i < numRows; i++) {
                                TblSetRowStaticHeight(table, i, true);
                                }
                                //TblSetLoadDataProcedure(table, 1, LoadTextTableItem);
                                for (i = 0; i < numTableColumns; i++) {
```

```
                                        TblSetColumnUsable(table, i, true);
                                        TblSetColumnSpacing(table, i, 0);
                                }
                                TblSetRowUsable(table, 1, false);
                                TblSetRowUsable(table, 2, false);
                                TblSetRowUsable(table, 3, false);
                                TblSetRowUsable(table, 4, false);
                                break;

                        case 6: //list + txt


                                if (blptr) {
                                        for (i=0; i<blptrsize; i++) {
                                                if(blptr[i]) MemPtrFree(blptr[i]);
                                                }
                                        MemPtrFree(blptr);
                                }

                                blptr=(char **)MemPtrNew(qni*sizeof(char *));
                                blptrsize=qni;

                                //int i;
                                //ListType* l;

                                l  =  (ListType*)FrmGetObjectPtr(form,  FrmGetObjectIndex(form,
1208));
                                /* ... */
                                //teraz wypelnienie tablicy wskaznikow (blptr) wartosciami
                                for (i=0; i<qni; i++) { //ilePozycji
                                //odczyt odpowiedzi z bazy
                                h=DmQueryRecord(gLibDB, DBqindex+2+i);
                                if (h) {
                                        MyRecordType *p=(MyRecordType *)MemHandleLock(h);
                                        //odczyt rekordu
                                        blptr[i]=(char *)MemPtrNew(30);

                                        StrCopy(blptr[i], &p->txt[8]);
                                        //FrmCustomAlert(InfoAlert, &p->txt[8], NULL, NULL);
                                        MemPtrUnlock(p);
                                        }
                                }

                                LstSetSelection(l, 0);
                                LstSetListChoices(l, blptr, qni);
                                LstSetHeight(l, qni);


                                TblSetItemStyle(table, 0, 0, labelTableItem);
                                TblSetItemPtr(table, 0, 0, "->");
                                TblSetItemStyle(table, 1, 0, labelTableItem);
                                TblSetItemPtr(table, 1, 0, "Inne");

                                TblSetItemStyle(table, 0, 1, popupTriggerTableItem);
                                TblSetItemInt(table, 0, 1, 0);
```

```
                                TblSetItemPtr(table, 0, 1, l);
                                TblSetItemStyle(table, 1, 1, textTableItem);

                                for (i = 0; i < numRows; i++) {
                                TblSetRowStaticHeight(table, i, true);
                                }
                                //TblSetLoadDataProcedure(table, 1, LoadTextTableItem);
                                for (i = 0; i < numTableColumns; i++) {
                                        TblSetColumnUsable(table, i, true);
                                        TblSetColumnSpacing(table, i, 0);
                                }
                                TblSetRowUsable(table, 1, true);
                                TblSetRowUsable(table, 2, false);
                                TblSetRowUsable(table, 3, false);
                                TblSetRowUsable(table, 4, false);
                                break;

                case 7: //txt

                                TblSetItemStyle(table, 0, 0, labelTableItem);
                                TblSetItemPtr(table, 0, 0, "Odp");
                                TblSetItemStyle(table, 0, 1, textTableItem);
                                TblSetRowUsable(table, 0, true);
                                for (i=1; i < numRows; i++) {
                                TblSetRowUsable(table, i, false);
                                }

                                for (i = 0; i <= numTableColumns; i++) {
                                        TblSetColumnUsable(table, i, true);
                                        TblSetColumnSpacing(table, i, 0);
                                }

                                break;
                default:
                                for (i = 0; i < numTableColumns; i++) {
                                        TblSetColumnUsable(table, i, false);
                                }
                                break;
        }

        TblSetLoadDataProcedure(table, 1, LoadTextTableItem);
        TblSetSaveDataProcedure(table, 1, SaveTextTableItem);
        TblDrawTable(table);
        FrmDrawForm(form);
}

static void UstawPyt(Int32 nrpyt)
{
//odczyt pytania z bazy
        MemHandle h;
        char qparam[5];
        char qbuf[50];
        UInt16 qtyp;

        h=DmQueryRecord(gLibDB, DBqindex);
```

```
        if (h) {
                MyRecordType *p=(MyRecordType *)MemHandleLock(h);
                //odczyt rekordu
                StrCopy(qparam, &p->txt[8]);
                //FrmCustomAlert(InfoAlert, &p->txt[8], NULL, NULL);
                MemPtrUnlock(p);
                }

        h=DmQueryRecord(gLibDB, DBqindex+1);
        if (h) {
                MyRecordType *p=(MyRecordType *)MemHandleLock(h);
                //odczyt rekordu
                StrCopy(qbuf, &p->txt[8]);
                //FrmCustomAlert(InfoAlert, &p->txt[8], NULL, NULL);
                MemPtrUnlock(p);
                }
//Przesuniecie odpowiedzi
        char qn[5];
        UInt16 qni=StrAToI(StrCopy(qn, &qparam[2]));
        DBqnext=2+qni;



FormPtr frmP;
frmP = FrmGetActiveForm();

FieldPtr nrpytf,pytf,odpf;

nrpytf=(FieldPtr) FrmGetObjectPtr(frmP,FrmGetObjectIndex(frmP,1202));
pytf=(FieldPtr) FrmGetObjectPtr(frmP,FrmGetObjectIndex(frmP,1203));
odpf=(FieldPtr) FrmGetObjectPtr(frmP,FrmGetObjectIndex(frmP,1207));

char *npyttp=FldGetTextPtr(nrpytf);
char *pyttp=FldGetTextPtr(pytf);
char nrpytt[10], pytt[255];

StrIToA(nrpytt,nrpyt);
FldSetTextHandle(nrpytf, 0);
FldInsert(nrpytf,nrpytt,StrLen(nrpytt));
InicjacjaTablicy(frmP, 1);

//ustawienie pytania i odpowiedzi
//ustalenie typu pytania
qtyp=0;
if (qparam[0]=='c' && qparam[1]=='0') qtyp=1;
if (qparam[0]=='c' && qparam[1]=='1') qtyp=2;
if (qparam[0]=='t' && qparam[1]=='0') qtyp=3;
if (qparam[0]=='t' && qparam[1]=='1') qtyp=4;
if (qparam[0]=='l' && qparam[1]=='0') qtyp=5;
if (qparam[0]=='l' && qparam[1]=='1') qtyp=6;
if (qparam[0]=='x') qtyp=7;

//zabezpieczenie przed za duza iloscia odpowiedzi
if (qtyp==1&&qni>5) qtyp=0;
if (qtyp==2&&qni>4) qtyp=0;
```

```
if (qtyp==3&&qni>5) qtyp=0;
if (qtyp==4&&qni>4) qtyp=0;
if (qtyp==5&&qni>5) qtyp=0;
if (qtyp==6&&qni>5) qtyp=0;
if (qtyp==7&&qni>0) qtyp=0;

//ustawienie pytania i odpowiedzi
aqtyp=qtyp; //globalizacja
aqni=qni;       //globalizacja

if (qtyp>0&&qtyp<8) {
        StrCopy(pytt, qbuf);
        FldSetTextHandle(pytf, 0);
        FldInsert(pytf, pytt,StrLen(pytt));
        InicjacjaTablicy(frmP, qtyp, qni);
        }
        else {
        StrCopy(pytt, "Blad w tresci pytania!!!");
        FldSetTextHandle(pytf, 0);
        FldInsert(pytf, pytt,StrLen(pytt));
        InicjacjaTablicy(frmP);
        }

}
static char * NullGetter_GetString(void * data, UInt16 fieldNum) {
    char * * fields = (char **)data;

    return fields[fieldNum];
}

static UInt32
MobileDB_PackedSize(DataSourceGetter * getter, void * data, UInt16 length)
{
    UInt32 size;
    UInt16 i;
    char * str;

    /* The header is the 7 byte sequence: 0xFF,0xFF,0xFF,0x01,0xFF,0x00,0x00 */
    size = 7;

    /* Each field is a 1 byte field #, the string itself, and the null byte. */
    for (i = 0; i < length; ++i) {
        str = getter->GetString(data, i);
        size += 1 + StrLen(str) + 1;
    }

    /* The trailer is the single byte 0xFF. */
    size += 1;

    return size;
}

static void
MobileDB_PackRecord(void * dest,
                    DataSourceGetter * getter, void * data, UInt16 length)
```

```
{
    UInt8 header[] = { 0xFF, 0xFF, 0xFF, 0x01, 0xFF, 0x00, 0x00 };
    UInt32 offset;
    UInt8 i;
    char * str;

    /* Pack the record header into position. */
    DmWrite(dest, 0, &header[0], sizeof(header));

    /* Pack all of the fields into the record. */
    offset = 7;
    for (i = 0; i < length; ++i) {
        /* Pack the current field number into the record. */
        DmWrite(dest, offset, &i, sizeof(i));
        offset += 1;

        /* Pack the string into the record. */
        str = getter->GetString(data, i);
        DmWrite(dest, offset, str, StrLen(str) + 1);
        offset += StrLen(str) + 1;
    }

    /* Pack the trailer into the record. */
    i = 0xFF;
    DmWrite(dest, offset, &i, sizeof(i));
}

static void MakeMetaRecord(DmOpenRef db, UInt8 category, char ** fields, UInt16 length) {

    UInt16 index = dmMaxRecordIndex;
    MemHandle h;
    UInt32 size;
    UInt16 attr;
    DataSourceGetter getter;

    getter.GetString = NullGetter_GetString;
    size = MobileDB_PackedSize(&getter, fields, length);
    h = DmNewRecord(db, &index, size);
    MobileDB_PackRecord(MemHandleLock(h), &getter, fields, length);
    MemHandleUnlock(h);
    DmReleaseRecord(db, index, true);

    /* Update the category so it is marked as a metadata record. */
    DmRecordInfo(db, index, &attr, 0, 0);
    attr &= ~(dmRecAttrCategoryMask);
    attr |= category;
    DmSetRecordInfo(db, index, &attr, 0);
}

static void InitAppInfoBlock(DmOpenRef db) {

    MemHandle appInfoHand;
    LocalID appInfoID, dbID;
    MobileDBInfoType info;
    UInt16 i, cardNo;
```

```
    void * p;


    /* Clear out the entire block. */
    MemSet(&info, sizeof(info), 0);

    /* Fill in the block. */
    info.renamedCategories = 0;
    StrCopy(info.categoryLabels[0], "Unfiled");
    StrCopy(info.categoryLabels[1], "FieldLabels");
    StrCopy(info.categoryLabels[2], "DataRecords");
    StrCopy(info.categoryLabels[3], "DataRecordsFout");
    StrCopy(info.categoryLabels[4], "Preferences");
    StrCopy(info.categoryLabels[5], "DataType");
    StrCopy(info.categoryLabels[6], "FieldLengths");
    info.lastUniqID = 15;
    info.version = 1;
    for (i = 0; i < 3; ++i) {
        info.filter[i].fieldNo = -1;
        info.sort[i].fieldNo = -1;
    }

    /* Allocate and fill a handle for app info block. */
    appInfoHand = DmNewHandle(db, sizeof(info));
    p = MemHandleLock(appInfoHand);
    DmWrite(p, 0, &info, sizeof(info));
    MemPtrUnlock(p);

    /* Set the app info block into place. */
    DmOpenDatabaseInfo(db, &dbID, 0, 0, &cardNo, 0);
    appInfoID = MemHandleToLocalID(appInfoHand);
    DmSetDatabaseInfo(cardNo, dbID, 0, 0, 0, 0, 0, 0, 0, &appInfoID, 0, 0, 0);
}


static Err MobileDB_Create(const char* title) {
    Err err;
    LocalID dbID;
    DmOpenRef db;
    char * fields[MAX_FIELDS];
    UInt16 i;

    err = DmCreateDatabase(0, title, libCreatorIDa, libDBTypea, false);
    if (err)
        return err;

    dbID = DmFindDatabase(0, title);
    if (!dbID)
        return DmGetLastErr();

    db = DmOpenDatabase(0, dbID, dmModeReadWrite);
    if (!db)
        return DmGetLastErr();

    /* Build the application information block. */
```

```
    InitAppInfoBlock(db);

    /* Create the field names record. */
    //for (i = 0; i < 7; ++i) fields[i] = &NameOdp[i];
    MakeMetaRecord(db, CATEGORY_FIELD_LABELS, NameOdp, 7);

    /* Create the field types record. */
    for (i = 0; i < MAX_FIELDS; ++i) fields[i] = "T";
    MakeMetaRecord(db, CATEGORY_DATA_TYPES, fields, MAX_FIELDS);

    /* Create the widths record. */
    for (i = 0; i < MAX_FIELDS; ++i) fields[i] = "80";
    MakeMetaRecord(db, CATEGORY_FIELD_LENGTHS, fields, MAX_FIELDS);

    /* Create the preferences record. */
    for (i = 0; i < MAX_FIELDS; ++i) fields[i] = "\0x01";
    MakeMetaRecord(db, CATEGORY_PREFERENCES, fields, MAX_FIELDS);

    /* Close the database. */
    DmCloseDatabase(db);


    return 0;
}
static void ZapiszAnkiete()
{
//rozbiegowka rekordu
char recs[8]={'\xff', '\xff', '\xff', '\x01', '\xff', '\x00', '\x00', '\x00'};
//koniec rekordu
char rece='\xff';

//ustawienie trybu bazy
UInt16 mode;
mode = dmModeReadWrite | dmModeShowSecret;

//otwarcie bazy danych
if (DmFindDatabase(0, libDBNamea)) gLibDBa =DmOpenDatabase(0,DmFindDatabase(0, libDBNamea),
mode);
if (! gLibDBa) {
        //tworzenie nowej bazy odpowiedzi
        MobileDB_Create(libDBNamea);
        if   (DmFindDatabase(0,   libDBNamea))   gLibDBa   =DmOpenDatabase(0,DmFindDatabase(0,
libDBNamea), mode);
        FrmCustomAlert(InfoAlert, "Stworzono nowa baze odpowiedzi.", NULL, NULL);
        }

if (! gLibDBa) FrmCustomAlert(InfoAlert, "Nie mozna otworzyc bazy odpowiedzi.", NULL, NULL);
        else {

        char buf[1000];
        UInt16 i,x,n;

        //zapis nr ankietowanego ale jest Zle!!!
        if (DmNumRecords(gLibDBa)<5) {
                for (n=0; n<8; n++) StrCopy(loaptr[n]->NrAnk, "1");
```

```
                }
                else {

                char qn[10];
                MemHandle h;

                h=DmQueryRecord(gLibDBa, DmNumRecords(gLibDBa)-1);

                for (n=0; n<loaptrsize; n++) {
                        //odczyt z bazy
                        if (h) {
                                MyRecordType *p=(MyRecordType *)MemHandleLock(h);
                                //odczyt rekordu
                                MemPtrUnlock(p);
                                StrCopy(loaptr[n]->NrAnk, StrIToA(qn,
                                        (StrAToI(&p->txt[8+StrLen(&p->txt[8])+2+StrLen(&p-
>txt[8+StrLen(&p->txt[8])+2])+2])
                                        +1)
                                        )
                                );
                                //FrmCustomAlert(InfoAlert, &p->txt[8], NULL, NULL);
                                //FrmCustomAlert(InfoAlert,    &p->txt[8+StrLen(&p->txt[8])+2],
NULL, NULL);
                                //FrmCustomAlert(InfoAlert,               &p->txt[8+StrLen(&p-
>txt[8])+2+StrLen(&p->txt[8+StrLen(&p->txt[8])+2])+2], NULL, NULL);

                                }

                        }
                }

        for (n=0;n<loaptrsize;n++) {

                //zapis nr rekordu
                if (DmNumRecords(gLibDBa)<5) {
                        StrCopy(loaptr[n]->Lp, "1");
                        }
                else {
                        char qn[10];
                        MemHandle h;
                        //odczyt z bazy
                        h=DmQueryRecord(gLibDBa, DmNumRecords(gLibDBa)-1);
                        if (h) {
                                MyRecordType *p=(MyRecordType *)MemHandleLock(h);
                                //odczyt rekordu
                                //FrmCustomAlert(InfoAlert, &p->txt[8], NULL, NULL);
                                MemPtrUnlock(p);
                                StrCopy(loaptr[n]->Lp, StrIToA(qn, StrAToI(&p->txt[8])+1));
                        }

                        }



                for(i=0, x=0;x<8;i++,x++) buf[i]=recs[x];
```

```
                for(x=0;x<StrLen(loaptr[n]->Lp);i++,x++)  buf[i]=loaptr[n]->Lp[x];  buf[i++]=0;
buf[i++]=1;
                for(x=0;x<StrLen(loaptr[n]->Owner);i++,x++)        buf[i]=loaptr[n]->Owner[x];
buf[i++]=0; buf[i++]=2;
                for(x=0;x<StrLen(loaptr[n]->NrAnk);i++,x++)        buf[i]=loaptr[n]->NrAnk[x];
buf[i++]=0; buf[i++]=3;
                for(x=0;x<StrLen(loaptr[n]->Data);i++,x++)          buf[i]=loaptr[n]->Data[x];
buf[i++]=0; buf[i++]=4;
                for(x=0;x<StrLen(loaptr[n]->Time);i++,x++)          buf[i]=loaptr[n]->Time[x];
buf[i++]=0; buf[i++]=5;
                for(x=0;x<StrLen(loaptr[n]->Pyt);i++,x++)            buf[i]=loaptr[n]->Pyt[x];
buf[i++]=0; buf[i++]=6;
                for(x=0;x<StrLen(loaptr[n]->Odp);i++,x++)            buf[i]=loaptr[n]->Odp[x];
buf[i++]=0;
                buf[i++]=rece;

                //rozmiar rekordu
                UInt16 sizer=i;//8+StrLen(buft)+1+1;

                //dodanie rekordu
                MemHandle memH;
                Char *p;
                UInt16 recindex = dmMaxRecordIndex;
                // Insert code for ADDREC
                memH = DmNewRecord(gLibDBa,&recindex,sizer);
                if (memH == 0)  FrmCustomAlert(InfoAlert,"Failed    to    add    new    record
:(",NULL,NULL);
                        else {
                        p = (Char*)MemHandleLock(memH);
                        if (p == NULL)  FrmCustomAlert(InfoAlert,"Failed    to    lock    down
pointer",NULL,NULL);
                                else {
                                //FrmCustomAlert(InfoAlert,buf,NULL,NULL);
                                MemSet(buf, 0x00, sizer);
                                DmWrite(p, 0, buf, sizer);
                                MemPtrUnlock(p);
                                DmReleaseRecord(gLibDBa,recindex,true);

                                /* Update the category so it is marked as a data record. */
                            UInt16 attr;
                            DmRecordInfo(gLibDBa, recindex, &attr, 0, 0);
                                attr &= ~(dmRecAttrCategoryMask);
                        attr |= CATEGORY_DATA_RECORDS & dmRecAttrCategoryMask;
                        DmSetRecordInfo(gLibDBa, recindex, &attr, 0);

                                }
                        }
        }
        //zamkniecie bazy
        DmCloseDatabase(gLibDBa);
        }

}
```

```
/***********************************************************************
 *
 * FUNCTION:    MainFormHandleEvent
 *
 * DESCRIPTION: This routine is the event handler for the
 *              "MainForm" of this application.
 *
 * PARAMETERS:  eventP  - a pointer to an EventType structure
 *
 * RETURNED:    true if the event has handle and should not be passed
 *              to a higher level handler.
 *
 * REVISION HISTORY:
 *
 *
 ***********************************************************************/
static Boolean MainFormHandleEvent(EventPtr eventP)
{
   Boolean handled = false;
   FormPtr frmP;



        switch (eventP->eType)
                {
                case menuEvent:
                        return MainFormDoCommand(eventP->data.menu.itemID);

                case frmOpenEvent:
                        frmP = FrmGetActiveForm();
                        MainFormInit( frmP);
                        FrmDrawForm ( frmP);
                        handled = true;
                        break;

                case ctlSelectEvent:
                        switch (eventP->data.ctlSelect.controlID)
                                {
                                case 1001: //rozpocznij ankiete

                                //ustawienie trybu bazy
                                UInt16 mode;
                                mode = dmModeReadWrite | dmModeShowSecret;

                                //otwarcie bazy danych
                                if         (DmFindDatabase(0,         libDBName))         gLibDB
=DmOpenDatabase(0,DmFindDatabase(0, libDBName), mode);
                                if (! gLibDB) {
                                        FrmCustomAlert(InfoAlert,  "Nie  mozna  otworzyc  bazy
pytan.", NULL, NULL);
                                        break;
                                        }

                                nrpyt=1;
```

```
                                DBqindex=DBqIndexStart;
                                MemHandle h;
                                char qparam[5];
                                char qn[5];
                                h=DmQueryRecord(gLibDB, DBqIndexStart-1);
                                        if (h) {
                                                MyRecordType                    *p=(MyRecordType
*)MemHandleLock(h);
                                                //odczyt rekordu
                                                StrCopy(qparam, &p->txt[8]);
                                                //FrmCustomAlert(InfoAlert,  &p->txt[8],  NULL,
NULL);
                                                MemPtrUnlock(p);
                                                }

                                DBqend=StrAToI(StrCopy(qn, &qparam[0]));

                                //lista odpowiedzi ankiety
                                loaptrsize=DBqend;
                                if (loaptr) {
                                        UInt16 i;
                                        for (i=0; i<loaptrsize; i++) {
                                                if(loaptr[i]) MemPtrFree(loaptr[i]);
                                                }
                                        MemPtrFree(loaptr);
                                }
                                loaptr=(MyOdpType      **)MemPtrNew(loaptrsize*sizeof(MyOdpType
*));

                                //teraz wypelnienie tablicy wskaznikow (loaptr) wartosciami
                                if (loaptr) {
                                        for (UInt16 i=0; i<loaptrsize; i++) { //ilePozycji
                                                loaptr[i]=(MyOdpType
*)MemPtrNew(sizeof(MyOdpType));
                                        }
                                }


                                FrmGotoForm(PytanieForm);
                                break;

                                case 1002: //zakoncz program
                                apruning=false;
                                break;

                                default:
                                break;
                                }

                default:
                        break;

                }

        return handled;
```

```
}

static Boolean PytanieFormHandleEvent(EventPtr eventP)
{
   Boolean handled = false;
   FormPtr frmP;
   MemHandle h;

        switch (eventP->eType)
                {
                case menuEvent:
                        return MainFormDoCommand(eventP->data.menu.itemID);

                case frmOpenEvent:
                        frmP = FrmGetActiveForm();
                        MainFormInit( frmP);
                        UstawPyt(nrpyt);
                        FrmDrawForm ( frmP);
                        handled = true;
                        break;

                case ctlSelectEvent:
                        switch (eventP->data.ctlSelect.controlID)
                                {
                                case 1204: //nastepne pytanie
                                DateTimeType now;
                                Char datenow[10], timenow[10];
                                Char nameBufP[50];

                                //wyciagniecie Ownera palmptopa
                                DlkGetSyncInfo(NULL, NULL, NULL, nameBufP, NULL, NULL);

                                //data i godzina
                                TimSecondsToDateTime(TimGetSeconds(), &now);
                                DateToAscii   (now.month,   now.day,   now.year,   dfDMYWithDots,
datenow);
                                TimeToAscii (now.hour, now.minute,tfDot24h, timenow);

                                //zapisz odpowiedz
                                //StrCopy(loaptr[nrpyt-1]->Lp, "1");
                                StrCopy(loaptr[nrpyt-1]->Owner, nameBufP);
                                //StrCopy(loaptr[nrpyt-1]->NrAnk, "100");
                                StrCopy(loaptr[nrpyt-1]->Data, datenow);
                                StrCopy(loaptr[nrpyt-1]->Time, timenow);
                                StrCopy(loaptr[nrpyt-1]->Odp, "");
                                //odczyt pytania z bazy
                                //MemHandle h;
                                h=DmQueryRecord(gLibDB, DBqindex+1);
                                if (h) {
                                        MyRecordType *p=(MyRecordType *)MemHandleLock(h);
                                        //odczyt rekordu
                                        StrCopy(loaptr[nrpyt-1]->Pyt, &p->txt[8]);
                                        MemPtrUnlock(p);
                                }
                                //zapis odpowiedzi
```

```
                                FormType *form = FrmGetActiveForm();
                                TablePtr    table    =    (TablePtr)    FrmGetObjectPtr(form,
FrmGetObjectIndex(form, 1207));


                                switch(aqtyp) {

                                        case 1: //checkbox
                                                        for (int i=0;i<aqni;i++) {
                                                                if    (TblGetItemInt(table,    i,
0)==1) {
                                                                        h=DmQueryRecord(gLibDB,
DBqindex+2+i);
                                                                        if (h) {
                                                                                MyRecordType
*p=(MyRecordType *)MemHandleLock(h);
                                                                                //odczyt
rekordu

        StrAdd(loaptr[nrpyt-1]->Odp, &p->txt[8]);

        StrAdd(loaptr[nrpyt-1]->Odp, "; ");

        MemPtrUnlock(p);
                                                                                }
                                                                }

                                                        }
                                                        break;

                                        case 2: //checkbox + txt
                                                        for (int i=0;i<aqni;i++) {
                                                                if    (TblGetItemInt(table,    i,
0)==1) {
                                                                        h=DmQueryRecord(gLibDB,
DBqindex+2+i);
                                                                        if (h) {
                                                                                MyRecordType
*p=(MyRecordType *)MemHandleLock(h);
                                                                                //odczyt
rekordu

        StrAdd(loaptr[nrpyt-1]->Odp, &p->txt[8]);

        StrAdd(loaptr[nrpyt-1]->Odp, "; ");

        MemPtrUnlock(p);
                                                                                }
                                                                }

                                                        }
                                                        if (StrLen(OdpStr)) {
                                                                StrAdd(loaptr[nrpyt-1]->Odp,
OdpStr);
```

```
                                                        StrAdd(loaptr[nrpyt-1]->Odp, ";
");
                                                        }
                                                break;

                                        case 3: //triger
                                                for (int i=0;i<aqni;i++) {
                                                        if   (TblGetItemInt(table,   i,
0)==1) {
                                                          h=DmQueryRecord(gLibDB,
DBqindex+2+i);
                                                        if (h) {
                                                                MyRecordType
*p=(MyRecordType *)MemHandleLock(h);
                                                                //odczyt
rekordu

        StrAdd(loaptr[nrpyt-1]->Odp, &p->txt[8]);

        StrAdd(loaptr[nrpyt-1]->Odp, "; ");

        MemPtrUnlock(p);
                                                                }
                                                        }

                                                }
                                                break;

                                        case 4: //triger + txt
                                                for (int i=0;i<aqni;i++) {
                                                        if   (TblGetItemInt(table,   i,
0)==1) {
                                                          h=DmQueryRecord(gLibDB,
DBqindex+2+i);
                                                        if (h) {
                                                                MyRecordType
*p=(MyRecordType *)MemHandleLock(h);
                                                                //odczyt
rekordu

        StrAdd(loaptr[nrpyt-1]->Odp, &p->txt[8]);

        StrAdd(loaptr[nrpyt-1]->Odp, "; ");

        MemPtrUnlock(p);
                                                                }
                                                        }

                                                }
                                                if (StrLen(OdpStr)) {
                                                        StrAdd(loaptr[nrpyt-1]->Odp,
OdpStr);
                                                        StrAdd(loaptr[nrpyt-1]->Odp, ";
");
                                                        }
```

```
                                                        break;

                                        case 5: //list
                                                        h=DmQueryRecord(gLibDB,
DBqindex+2+TblGetItemInt(table, 0, 1));
                                                        if (h) {
                                                                MyRecordType   *p=(MyRecordType
*)MemHandleLock(h);
                                                                //odczyt rekordu
                                                                StrAdd(loaptr[nrpyt-1]->Odp,
&p->txt[8]);
                                                                StrAdd(loaptr[nrpyt-1]->Odp, ";
");
                                                                MemPtrUnlock(p);
                                                        }
                                                        break;

                                        case 6: //list + txt
                                                        h=DmQueryRecord(gLibDB,
DBqindex+2+TblGetItemInt(table, 0, 1));
                                                        if (h) {
                                                                MyRecordType   *p=(MyRecordType
*)MemHandleLock(h);
                                                                //odczyt rekordu
                                                                StrAdd(loaptr[nrpyt-1]->Odp,
&p->txt[8]);
                                                                StrAdd(loaptr[nrpyt-1]->Odp, ";
");
                                                                MemPtrUnlock(p);
                                                        }
                                                        if (StrLen(OdpStr)) {
                                                                StrAdd(loaptr[nrpyt-1]->Odp,
OdpStr);
                                                                StrAdd(loaptr[nrpyt-1]->Odp, ";
");
                                                                }
                                                        break;

                                        case 7: //txt
                                                        if (StrLen(OdpStr)) {
                                                                StrAdd(loaptr[nrpyt-1]->Odp,
OdpStr);
                                                                StrAdd(loaptr[nrpyt-1]->Odp, ";
");
                                                                }
                                                        break;

                                        default:
                                                        StrCopy(loaptr[nrpyt-1]->Odp,   "Bledne
pytanie");
                                                        break;
                                }


                                nrpyt++;
```

```
                        if (nrpyt==(DBqend+1)) FrmGotoForm(KoniecAnkietyForm);
                                else {
                                        DBqindex=DBqindex+DBqnext;
                                        UstawPyt(nrpyt);
                                        }
                        break;

                        case 1205: //poprzednie pytanie
                        nrpyt--;
                        if (nrpyt==0) nrpyt=1;
                                //MemHandle h;
                                char qparam[5];
                                UInt16 i;
                                char qn[5];
                                DBqindex=DBqIndexStart;
                                for(i=1;i<nrpyt;i++) {
                                h=DmQueryRecord(gLibDB, DBqindex);

                                        if (h) {
                                                MyRecordType            *p=(MyRecordType
*)MemHandleLock(h);

                                                //odczyt rekordu
                                                StrCopy(qparam, &p->txt[8]);
                                                //FrmCustomAlert(InfoAlert, &p->txt[8],
NULL, NULL);
                                                MemPtrUnlock(p);
                                                }

                                        DBqindex=DBqindex+2+StrAToI(StrCopy(qn, &qparam[2]));
                                                }
                        UstawPyt(nrpyt);
                        break;

                        case 1206: //przerwanie ankiety

                        //zamkniecie bazy
                        if (gLibDB) DmCloseDatabase(gLibDB);

                        FrmGotoForm(MainForm);
                        break;

                        default:
                        break;
                        }


                case tblSelectEvent://tblEnterEvent:
                        {
                                Int16 row = eventP->data.tblEnter.row;
                                Int16 column = eventP->data.tblEnter.column;

                                if ((aqtyp==3||aqtyp==4)&&column==0) {


                        /*
```

```
                        char al[50], ab[10];
                        StrCopy(al, "column=");
                        StrAdd(al, StrIToA(ab, column));
                        StrAdd(al, " row=");
                        StrAdd(al, StrIToA(ab, row));
                        FrmCustomAlert(InfoAlert, al, NULL, NULL);
                        */
                                        FormType *form = FrmGetActiveForm();
                                        TablePtr  table  =  (TablePtr)  FrmGetObjectPtr(form,
FrmGetObjectIndex(form, 1207));

                                        int i;
                                        for        (i=0;        TblGetItemInt(table,        i,
0)==0&&i<aqni;i++){};
                                        if (i<aqni) {
                                                for  (i=0;  i<aqni;i++)  TblSetItemInt(table,  i,
0, 0);
                                                TblSetItemInt(table, row, 0, 1);
                                                for  (i=0;  i<aqni;i++)  TblMarkRowInvalid(table,
i);
                                                TblRedrawTable(table);
                                                }
                                        }

                        }
                        break;
                default:
                        break;

                }

        return handled;
}

static Boolean KoniecAnkietyFormHandleEvent(EventPtr eventP)
{
   Boolean handled = false;
   FormPtr frmP;

        switch (eventP->eType)
                {
                case menuEvent:
                        return MainFormDoCommand(eventP->data.menu.itemID);

                case frmOpenEvent:
                        frmP = FrmGetActiveForm();
                        MainFormInit( frmP);
                        FrmDrawForm ( frmP);
                        handled = true;
                        break;

                case ctlSelectEvent:
                        switch (eventP->data.ctlSelect.controlID)
                                {
                                case 1303: //zakonczenie ankiety
```

```
                                ZapiszAnkiete();

                                //zamkniecie bazy
                                if (gLibDB) DmCloseDatabase(gLibDB);

                                FrmGotoForm(MainForm);
                                break;

                                case 1302: //poprzednie pytanie
                                nrpyt--;
                                if (nrpyt==0) FrmGotoForm(MainForm);
                                        else FrmGotoForm(PytanieForm);
                                break;

                                case 1301: //przerwanie ankiety

                                //zamkniecie bazy
                                if (gLibDB) DmCloseDatabase(gLibDB);

                                FrmGotoForm(MainForm);
                                break;

                                default:
                                break;
                                }

                default:
                        break;

                }

        return handled;
}
/***********************************************************************
 *
 * FUNCTION:    AppHandleEvent
 *
 * DESCRIPTION: This routine loads form resources and set the event
 *              handler for the form loaded.
 *
 * PARAMETERS:  event  - a pointer to an EventType structure
 *
 * RETURNED:    true if the event has handle and should not be passed
 *              to a higher level handler.
 *
 * REVISION HISTORY:
 *
 *
 ***********************************************************************/
static Boolean AppHandleEvent(EventPtr eventP)
{
        UInt16 formId;
        FormPtr frmP;

        if (eventP->eType == frmLoadEvent)
```

```
                {
                // Load the form resource.
                formId = eventP->data.frmLoad.formID;
                frmP = FrmInitForm(formId);
                FrmSetActiveForm(frmP);

                // Set the event handler for the form.  The handler of the currently
                // active form is called by FrmHandleEvent each time is receives an
                // event.
                switch (formId)
                        {
                        case MainForm:
                                FrmSetEventHandler(frmP, MainFormHandleEvent);
                                break;
                        case PytanieForm:
                                FrmSetEventHandler(frmP, PytanieFormHandleEvent);
                                break;
                        case KoniecAnkietyForm:
                                FrmSetEventHandler(frmP, KoniecAnkietyFormHandleEvent);
                                break;
                        default:
//                              ErrFatalDisplay("Invalid Form Load Event");
                                break;

                        }
                return true;
                }

        return false;
}


/***********************************************************************
 *
 * FUNCTION:    AppEventLoop
 *
 * DESCRIPTION: This routine is the event loop for the application.
 *
 * PARAMETERS:  nothing
 *
 * RETURNED:    nothing
 *
 * REVISION HISTORY:
 *
 *
 ***********************************************************************/
static void AppEventLoop(void)
{
        UInt16 error;
        EventType event;

        do {
                EvtGetEvent(&event, evtWaitForever);

                if (! SysHandleEvent(&event))
```

```
                        if (! MenuHandleEvent(0, &event, &error))
                                if (! AppHandleEvent(&event))
                                        FrmDispatchEvent(&event);

        } while ((event.eType != appStopEvent)&&apruning);
}


/***********************************************************************
 *
 * FUNCTION:     AppStart
 *
 * DESCRIPTION:  Get the current application's preferences.
 *
 * PARAMETERS:   nothing
 *
 * RETURNED:     Err value 0 if nothing went wrong
 *
 * REVISION HISTORY:
 *
 *
 ***********************************************************************/
static Err AppStart(void)
{
    StarterPreferenceType prefs;
    UInt16 prefsSize;

        // Read the saved preferences / saved-state information.
        prefsSize = sizeof(StarterPreferenceType);
        if (PrefGetAppPreferences(appFileCreator, appPrefID, &prefs, &prefsSize, true) !=
                noPreferenceFound)
                {
                }



        ListaOdp=(char **)MemPtrNew(numTableRows*sizeof(char *));
        //teraz wypelnienie tablicy wskaznikow
        if (ListaOdp) {
                for (UInt16 i=0; i<numTableRows; i++) { //ilePozycji
                ListaOdp[i]=(char *)MemPtrNew(20*sizeof(char));
                                        }
                                }

        Int16  i, j;
        for (i = 0; i < numTextColumns; i++) {
                for (j = 0; j < numTableRows; j++) {
                        Char  *str;
                        gTextHandles[i][j] = MemHandleNew(20);
                        //ListaOdp[i][j] = (char**)MemPtrNew(20);

                        str = (char*)MemHandleLock(gTextHandles[i][j]);
                        *str = '\0';
                        MemHandleUnlock(gTextHandles[i][j]);
                }
```

```
        }

        return errNone;
}


/***********************************************************************
 *
 * FUNCTION:    AppStop
 *
 * DESCRIPTION: Save the current state of the application.
 *
 * PARAMETERS:  nothing
 *
 * RETURNED:    nothing
 *
 * REVISION HISTORY:
 *
 *
 ***********************************************************************/
static void AppStop(void)
{
   StarterPreferenceType prefs;
   Int16  i, j;


        for (i = 0; i < numTextColumns; i++) {
                for (j = 0; j < numTableRows; j++) {
                        MemHandleFree(gTextHandles[i][j]);
                        //MemPtrFree(ListaOdp[i][j]);
                }
        }

        // Write the saved preferences / saved-state information.  This data
        // will be backed up during a HotSync.
        PrefSetAppPreferences (appFileCreator, appPrefID, appPrefVersionNum,
                &prefs, sizeof (prefs), true);



        //zwolnienie list
        if (ListaOdp) {
                                        for (i=0; i<numTableRows; i++) {
                                                if(ListaOdp[i]) MemPtrFree(ListaOdp[i]);
                                                }
                                        MemPtrFree(ListaOdp);
                                }

        if (blptr) {
                                        for (i=0; i<blptrsize; i++) {
                                                if(blptr[i]) MemPtrFree(blptr[i]);
                                                }
                                        MemPtrFree(blptr);
                                }
```

```
        if (loaptr) {
                                        for (i=0; i<loaptrsize; i++) {
                                                if(loaptr[i]) MemPtrFree(loaptr[i]);
                                                }
                                        MemPtrFree(loaptr);
                                }
        // Close all the open forms.
        FrmCloseAllForms ();
}


/***********************************************************************
 *
 * FUNCTION:    StarterPalmMain
 *
 * DESCRIPTION: This is the main entry point for the application.
 *
 * PARAMETERS:  cmd - word value specifying the launch code.
 *              cmdPB - pointer to a structure that is associated with the launch code.
 *              launchFlags -  word value providing extra information about the launch.
 *
 * RETURNED:    Result of launch
 *
 * REVISION HISTORY:
 *
 *
 ***********************************************************************/
static UInt32 StarterPalmMain(UInt16 cmd, MemPtr /*cmdPBP*/, UInt16 launchFlags)
{
        Err error;

        error = RomVersionCompatible (ourMinVersion, launchFlags);
        if (error) return (error);

        switch (cmd)
                {
                case sysAppLaunchCmdNormalLaunch:
                        error = AppStart();
                        if (error)
                                return error;

                        FrmGotoForm(MainForm);
                        AppEventLoop();
                        AppStop();
                        break;

                default:
                        break;

                }

        return errNone;
}
```

```
/***********************************************************************
 *
 * FUNCTION:    PilotMain
 *
 * DESCRIPTION: This is the main entry point for the application.
 *
 * PARAMETERS:  cmd - word value specifying the launch code.
 *              cmdPB - pointer to a structure that is associated with the launch code.
 *              launchFlags -  word value providing extra information about the launch.
 * RETURNED:    Result of launch
 *
 * REVISION HISTORY:
 *
 *
 ***********************************************************************/
UInt32 PilotMain( UInt16 cmd, MemPtr cmdPBP, UInt16 launchFlags)
{
    return StarterPalmMain(cmd, cmdPBP, launchFlags);
}
```

Bibliografia

1. Lonnon R. Foster *Palm OS Programming Bible,* IDG Books Worldwide , 2000

2. Neil Rhodes i Julie *McKeehan Palm Programming - The Developers Guide,* O'Reilly*, 1998*

3. Dominik Oślizło *Palm, czyli co?,* www.palmpage.pl

4. Hubert Trzewik *Szybki wstęp do programowania PalmOS*, www.palmpage.pl

5. Hubert Trzewik *Szkielet aplikacji PalmOS,* www.palmpage.pl

6. Holger Klawitter *MobileDB Database File Structure*