Pola wymieniane za słowem SELECT mogą pochodzić z różnych powiązanych ze sobą tablic. Możliwości tej instrukcji odpowiadają poleceniom omawianym w rozdz. 6.4.3.

Pełna wersja instrukcji SELECT zawiera jeszcze sześć innych słów kluczowych języka SQL i towarzyszących im nazw tablic, kolumn lub instrukcji języka. Możliwości formułowania zapytań za pomocą tej instrukcji są duże. Są programy wspomagające formułowanie zapytań w języku SQL (np. MS Query).

Język SQL zawiera wiele innych instrukcji, funkcji, a także pojęć. Jeżeli zainteresowałeś się tym językiem, to sprawdź, czy w Twojej pracowni jest dostępny system zarządzania bazami danych, w którym można formułować zapytania w języku SQL (np. dBase IV 2.0, FoxPro 2.0 lub ich późniejsze wersje, Access 2.0) i porozum się z nauczycielem w sprawie możliwości dokładniejszego poznania tego języka.

11.3. Pliki wsadowe — programy

Pewne możliwości programowania stwarzają pliki typu *.BAT, czyli pliki przetwarzane w trybie wsadowym. Za ich pomocą możesz ułatwić sobie wykonywanie typowych czynności w systemie operacyjnym DOS. Zapisywane w nich polecenia systemu DOS są traktowane jak instrukcje programu, a całe pliki jak programy. Można też stosować w nich instrukcje, których nie używa się w trybie bezpośrednim i dzięki temu wykonywać pracochłonne zadania, a także zadania nietypowe. Co więcej, mogą one stanowić dogodne uzupełnienie programów użytkowanych i wykonanych przez Ciebie ułatwiając ich współpracę.

11.3.1. Podstawowe możliwości

Znasz już kilka elementów stosowanych w plikach typu *.BAT. Instrukcje ECHO OFF i ECHO ON, wyłączające i włączające powtarzanie poleceń systemu DOS na ekranie, mogą służyć do przekazywania użytkownikowi **komunikatów**. Komentarze pisane po słowie REM odgrywają rolę faktycznych komentarzy dla osoby oglądającej zawartość pliku, jeżeli podczas wykonywania programu nie są pokazywane na ekranie, ale jeżeli są widoczne, to pełnią rolę komunikatów do użytkownika.

Żeby użytkownik miał czas na przeczytanie komunikatu, powinieneś użyć instrukcji PAUSE, po której możesz umieścić tekst. Wykonywanie poleceń zawartych w pliku zostaje wtedy wstrzymane do czasu naciśnięcia jakiegoś klawisza, przy czym tekst zawarty za słowem PAUSE zostaje wyświetlony.

Jeżeli wśród instrukcji zawartych w pliku *.BAT byłoby zawarte polecenie drukowania, to w instrukcji PAUSE mógłbyś np. zamieścić komunikat dotyczący założenia papieru i potwierdzenia gotowości drukarki.

Program BAT-1

```
@ECHO OFF
REM plik BAT-1
ECHO ON
PAUSE Czy drukarka jest gotowa?
PRINT bat01.bat
REM Plik wyslany do drukarki
```

11.3.2. Parametry

Program taki jak BAT-1 mógłbyś chcieć stosować do plików o różnych nazwach. System DOS stwarza możliwość użycia parametrów. W poleceniu dla systemu DOS po nazwie pliku wymienia się parametry oddzielone od siebie co najmniej jedną spacją. W programie parametry te oznacza się symbolem procentu z jednocyfrowym numerem parametru, np. %2 oznacza drugi parametr. Zamień nazwę pliku na symbol pierwszego parametru i zapisz zmieniony plik BAT01.BAT jako BAT02.BAT.

Program BAT-2

```
@ECHO OFF
REM plik BAT-2
ECHO ON
PAUSE Czy drukarka jest gotowa?
PRINT %1
REM Plik wyslany do drukarki
```

Wywołaj program BAT-2 podając po nazwie pliku BAT02 nazwę pliku, który ma zostać wydrukowany, np.

```
bat02 wsad1.bat
```

Za pomocą odpowiedniego pliku typu *.BAT możesz wspomóc gromadzenie danych o plikach zapisanych na dyskietkach w celu wprowadzenia ich do bazy.

Przyjmij, że dysponujesz programem wykonywalnym PLIKKAT.EXE, który odczytuje plik AA sporządzony poleceniem DIR > AA i tworzy plik PLIKDIR zawierający dane przygotowane do wczytania do bazy danych. Napisz program wsadowy nadający dyskietce etykietę (poleceniem LABEL systemu DOS), tworzący plik AA, uruchamiający program PLIKKAT.EXE i zmieniają-

cy nazwę utworzonego przez ten program pliku PLIKDIR na nazwę składającą się z litery A i etykiety dyskietki oraz rozszerzenia TXT.

Program BAT-3

```
@ECHO OFF
REM plik BAT-3
ECHO ON
PAUSE Wloz dyskietke do napedu A:
LABEL A: %1
@ECHO OFF
REM tworzenie pliku AA poleceniem DIR
DIR a: /s > aa
REM przetwarzanie pliku AA na PLIKDIR
REM za pomoca programu PLIKKAT
plikkat
REM zmiana nazwy pliku PLIKDIR
REN plikdir a%1.txt
ECHO ON
```

11.3.3. Instrukcje warunkowe, instrukcje skoku i etykiety

System DOS umożliwia stosowanie w plikach typu *.BAT instrukcji warunkowych. Są trzy odmiany tych instrukcji i badanych w nich warunków:

- równość dwóch łańcuchów: IF łańcuch1 == łańcuch2 polecenie;
- istnienie pliku: IF EXIST nazwapliku polecenie (*jeżeli istnieje*);
- poziom błędu: IF ERRORLEVEL numer polecenie (*errorlevel — poziom błędu*).

Jest też dostępna instrukcja skoku: GOTO *etykieta*, do miejsca programu, w którym znajduje się etykieta, tj. nazwa poprzedzona znakiem dwukropka, zapisana w odrębnej linii.

Jednym z przykładowych zastosowań instrukcji z badaniem równości łańcuchów jest wybieranie wersji konfiguracji systemu operacyjnego, gdy ze względu na używane programy i sprzęt wygodnie jest operować kilkoma różnymi wersjami.

Załóżmy więc, że w katalogu C:\BAT\AUTOCONF masz przygotowane pliki AUTOEXEC i CONFIG z różnymi rozszerzeniami odpowiadającymi różnym zastosowaniom, np. DOS, 852 i WIN. Drugi wariant obejmuje uruchomienie strony kodowej 852 z polskimi znakami, czego wymagają niektóre programy, a trzeci uruchomienie środowiska Windows.

Przygotowanie komputera do pracy w innym wariancie polega na skopiowaniu odpowiedniej pary plików do katalogu głównego na dysku C ze zmianą nazwy na AUTOEXEC.BAT i CONFIG.SYS. Wobec częstego zmieniania

warunków pracy, chciałbyś czynności te wykonywać za pomocą programu wsadowego. Mógłbyś przygotować następujący program.

Program BAT-4

```
@echo off
if '%1'=='-?' goto brakwersji
if '%1'=='/?' goto brakwersji
if '%1'=='852' goto wersja852
if '%1'=='dos' goto wersjados
if '%1'=='DOS' goto wersjados
if '%1'=='win' goto wersjawin
if '%1'=='WIN' goto wersjawin
goto brakwersji
:wersjados
   copy c:\bat\autoconf\autoexec.dos c:\autoexec.bat
   copy c:\bat\autoconf\config.dos   c:\config.sys
   goto koniec
:wersja852
   copy c:\bat\autoconf\autoexec.852 c:\autoexec.bat
   copy c:\bat\autoconf\config.852   c:\config.sys
   goto koniec
:wersjawin
   copy c:\bat\autoconf\autoexec.win c:\autoexec.bat
   copy c:\bat\autoconf\config.win   c:\config.sys
   goto koniec
:brakwersji
   echo
   echo PRZYGOT kopiuje wybrane wersje
   echo plikow autoexec.bat i˜config.sys
   echo opcje PRZYGOT:  DOS 852 WIN
   echo /?  - wyswietla te informacje
:koniec
```

11.3.4. Pętle

Instrukcji warunkowych używa się także do tworzenia pętli. Możliwe jest na przykład wielokrotne wywoływanie nawzajem programów współpracujących ze sobą i komunikujących się poprzez pliki ASCII zapisywane przez jedne programy a odczytywane przez inne. Do przerwania takiej pętli może posłużyć instrukcja IF EXIST odnosząca się do nazwy pliku, który zostałby utworzony przez jeden ze współpracujących programów, kiedy byłyby spełnione warunki zakończenia całości.

Można też pewne polecenia wykonać dla wskazanej grupy plików posługując się instrukcją FOR.

W przypadku programów wsadowych ma ona postać

```
FOR zmienna IN (grupa plików) DO polecenie
```

przy czym nazwa zmiennej składa się z dwóch znaków % i jednej litery, np. %%A. Na przykład za pomocą programu BAT-5 możesz przedstawić na ekranie nazwy plików należących do jednej z trzech klas: *.BAT, *,COM, *.EXE.

Program BAT-5

```
ECHO OFF
FOR %%A IN (*.BAT *.COM *.EXE) DO dir %%A /B
```

11.3.5. Wywoływanie innych programów wsadowych

Podczas pracy interaktywnej programy zapisane w plikach typu *.BAT wywołuje się tak samo jak programy wykonywalne, tzn. poleceniem jest nazwa pliku. Podobne odwoływanie się wewnątrz programów wsadowych nie jest jednakowe. Programy wykonywalne są wywoływane nadal przez podanie nazwy pliku, natomiast programy zapisane w plikach typu *.BAT są wywoływane instrukcją CALL *nazwa* (*call* — wywołaj). Można powiedzieć, że wywoływane w ten sposób programy wsadowe są traktowane jak procedury.

11.4. Programowanie w środowisku Windows

11.4.1. Programy sterowane zdarzeniami

Programy wykonywalne działające w systemie DOS nie będą działać w środowisku Windows. Są wersje systemów programowania przeznaczone dla środowiska Windows; np. jest system Turbo Pascal dla Windows. Można w nim tworzyć programy, które będą działały w tym środowisku (natomiast nie będą działały bezpośrednio w systemie DOS).

Programy przeznaczone dla środowiska Windows uwzględniają w mniejszym lub większym stopniu specyfikę tego środowiska. Chodzi tu nie tylko o sprawy czysto informatyczne, jak np. gospodarka pamięcią komputera, lecz także, a nawet przede wszystkim o komunikację z użytkownikiem. Jeżeli posługiwałeś się zarówno programami dla systemu DOS, jak i dla środowiska Windows, to dostrzegasz różnicę.

W programach tworzonych w takich systemach jak QBasic i Turbo Pascal (dla DOS) podstawowe działania użytkownika z zakresu komunikacji z funkcjonującym programem polegają na wpisywaniu liczb i tekstów z klawiatury oraz na udzielaniu odpowiedzi typu tak lub nie. W programach

użytkowych dla DOS oprócz klawiatury używana jest coraz częściej także mysz. Komunikacja za pomocą myszy (lub podobnego urządzenia) nabiera jednak dużego znaczenia dopiero w środowisku Windows. Mysz służy tu do uaktywniania menu, wybierania i zatwierdzania poleceń z menu, wydawania poleceń przez „naciskanie" przycisków, uruchamiania programów, zmniejszania, zwiększania i przesuwania okien, przesuwania suwaków, przesuwania rozmaitych obiektów, uaktywniania okien (a więc i programów) i in.

Komunikacja z użytkownikiem wysuwa się tu na plan pierwszy, przed właściwe obliczenia. Konstruowane dla środowiska Windows programy określa się jako **sterowane zdarzeniami**. Zdarzeniami tymi są czynności wykonywane przez użytkownika za pomocą myszy oraz za pomocą klawiatury. Program czeka na akcje użytkownika i podejmuje działania zależnie od tego, co użytkownik uczynił.

Opracowywane ostatnio przez różne firmy pakiety do programowania w środowisku Windows reprezentują przedstawione podejście. Tworzenie programów przebiega w nich inaczej niż w systemie QBasic lub Turbo Pascal. Na pierwszy plan wysuwa się tworzenie ekranów, okien do komunikacji z użytkownikiem oraz rozróżnianie i obsługiwanie akcji użytkownika. Właściwe obliczenia czy przetwarzanie danych schodzą na plan drugi.

Przy pracy w środowisku graficznym Windows kontakt wzrokowy użytkownika z ekranem, na którym użytkownik operuje wskaźnikiem myszy, nabiera większego znaczenia niż w systemie DOS. Ten aspekt znajduje odbicie nawet w nazwach pakietów do programowania w środowisku Windows, w których występuje słowo *visual* (widzialny), na przykład Visual Basic, Visual Objects.

W tworzonych programach można zazwyczaj odwoływać się do istniejących innych programów i wykorzystywać je w pewnym stopniu. Na przykład można utworzyć program, dla którego część obliczeń zostanie wykonana w języku danego pakietu, część za pomocą dołączonego programu napisanego w innym języku (np. Turbo Pascal dla Windows), część prezentacyjna za pomocą arkusza kalkulacyjnego, a gromadzenie danych za pomocą programu zarządzania bazami danych. Tworzenie tego typu programów nie jest przy tym „sztuką dla sztuki", lecz wynika z ograniczenia pracy twórcy programu do tych części, które są niezbędne, przy posłużeniu się w takim stopniu programami gotowymi, w jakim jest to możliwe i celowe. Programowanie aplikacji jest dzięki takim możliwościom ułatwione.

Niektóre z opracowanych systemów programowania mają charakter uniwersalny (np. MS Visual Basic, Visual C++ oraz CA-Realizer), inne są zorientowane na pewne dziedziny, jak np. pakiet CA Visual Objects przeznaczony przede wszystkim do zarządzania relacyjnymi bazami danych.

Pewnego wyobrażenia o podejściu do programowania w środowisku Windows nabierzesz po zapoznaniu się z przykładem tworzenia programu w systemie Visual Basic.

11.4.2. Visual Basic

Omawianie systemu Visual Basic ograniczymy do niektórych najprostszych, a szczególnie charakterystycznych jego elementów.

Formy, moduły, projekty

Podstawowymi elementami, od których zaczynasz tworzenie programu, są **formy** (*form*). Forma jest to okno, które tworzysz do komunikowania się z programem. Umieszczasz na nim elementy graficzne, napisy i elementy sterujące. Opis tych elementów jest zapisywany w pliku o rozszerzeniu FRM.

Właściwy program w języku Visual Basic jest umieszczany w plikach zwanych **modułami** (*code module*); mają one rozszerzenie BAS. Są w nich zawarte procedury dotyczące reakcji na użycie przez użytkownika elementów sterujących. System przejmuje na siebie część zadań związanych z tworzeniem procedur, jak tworzenie nagłówków i nadawanie im nazw w taki sposób, że są one między sobą odpowiednio powiązane.

Uwaga. Program główny jest tworzony całkowicie przez system Visual Basic.

Zestaw plików, z których tworzysz aplikację (ostateczny program) określany jest mianem **projektu** (*project*). Projekt może zawierać wiele form i modułów. Plik zawierający informację na temat wszystkich elementów projektu ma rozszerzenie MAK.

W skład projektu wchodzą jeszcze elementy sterujące standardowe oraz określane przez użytkownika. Informacja o nich jest umieszczana w plikach o rozszerzeniu VBX. Pliki te są dołączane przez system.

Zrealizowany — nawet tylko częściowo — projekt możesz wykonać (poleceniem Run z menu). W razie trudności korzystaj z pomocy debugera.

Zrealizowany projekt możesz poddać kompilacji. W jej wyniku powstaje samodzielny program wykonywalny typu *.EXE. Do otrzymania takiego programu służy polecenie Make EXE File w menu File. Powstały program można następnie samodzielnie uruchomić (można to czynić nawet z systemu operacyjnego DOS).

Po ponownym uruchomieniu systemu Visual Basic możesz zacząć pracę z nowym projektem, a jeżeli już pracowałeś wcześniej, to możesz pracę kontynuować. Odpowiednikiem otwierania i zachowywania plików w innych programach użytkowych jest w systemie Visual Basic otwieranie projektów.

Postępowanie zilustrujemy przykładem. Po uruchomieniu systemu Visual Basic i ewentualnym wybraniu polecenia New Project (nowy projekt) z menu File, na ekranie ukazuje się okno formy opatrzone standardową nazwą Form1.

Na początek nadaj nazwę formie przez zapisanie jej poleceniem Save As. Ponieważ są to Twoje pierwsze kroki, użyjemy nazwy FRMWSTEP. (Poprzedzenie właściwej nazwy literami FRM sprawia, że wszystkie formy mogą być na liście plików skupione w jednym miejscu.) Od razu zapisz też projekt. Nie używaj standardowej nazwy PROJEKT1, lecz np. WSTEP.

Tworzenie formy

Możesz kontynuować edycję lub nawet zacząć od otworzenia zapisanego przez Ciebie projektu. Przywołujesz na ekran Toolbox, czyli „skrzynkę z narzędziami" (rys. 11.2). Liczba rodzajów dostępnych przycisków w skrzynce zależy od wersji systemu Visual Basic. Masz między innymi przyciski sterujące i okna tekstowe, którymi się posłużysz, a także przyciski opcji, suwaki pionowe i poziome, okna list, okna graficzne i wiele innych.

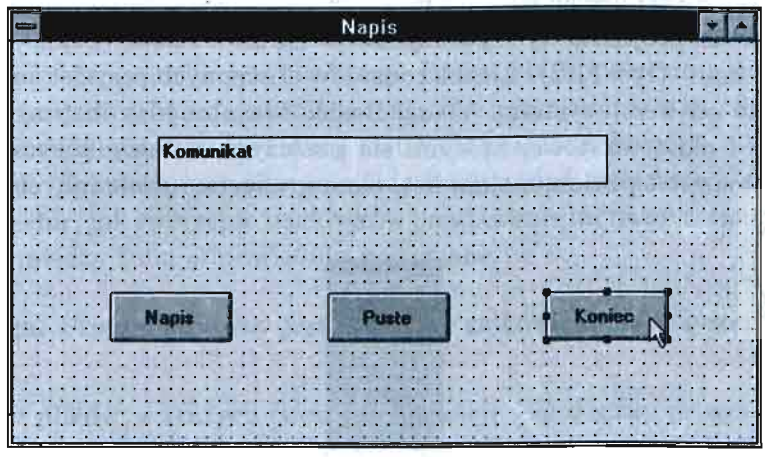
Rys. 11.2. Narzędzia do tworzenia formy w programie Visual Basic

Naciskając szybko dwukrotnie na dany przycisk ze skrzynki narzędziowej powodujesz ukazanie się w oknie formy wskaźnika w kształcie prostokąta, który następnie możesz myszą przesuwać we właściwe miejsce, a także rozciągać czy zwężać.

Tworzone okno ma zawierać cztery elementy: okno tekstowe (*Text Box*) i trzy przyciski sterujące (*Command Button*).

Po umiejscowieniu elementu powinieneś określić jego **cechy**. W tym celu wybierasz z menu polecenie Properties (cechy). Rodzaj cech zależy od rodzaju wybranego przycisku. Zawsze musisz nadać elementowi sterującemu **nazwę** (*name*). Ta nazwa jest potem używana w nazwach procedur, dzięki czemu wiadomo, do jakiego elementu dana procedura się odnosi. Ponadto możesz nadać elementowi **tytuł**. Jest to ta nazwa, która jest widoczna na ekranie, np. napis na przycisku (rys. 11.3).

Przyciskom sterującym nadaj nazwy i tytuły: „Napis", „Puste" i „Koniec". Oknu tekstowemu nadaj nazwę „Komunikat"; ponadto w polu Text okna Properties możesz określić tekst, który znajdzie się w oknie na początku działania programu, np. „Komunikat".



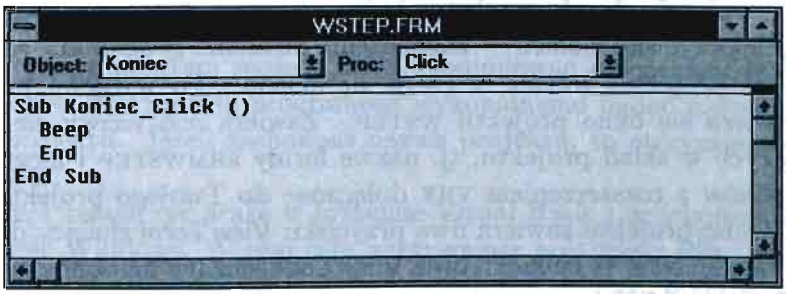
Rys. 11.3. Tworzenie formy w programie Visual Basic

Tworzenie modułów programu

Po utworzeniu i zapisaniu formy przystępujesz do tworzenia procedur obsługujących przyciski sterujące. Rozpoczynasz naciskając dwukrotnie przycisk. Otwiera się okno z nagłówkiem procedury obsługującej naciskanie danego przycisku. Swoje instrukcje wpisujesz wewnątrz procedury.

W przypadku przycisku Koniec wystarczy, jeżeli wpiszesz End (rys. 11.4).

Już po wpisaniu tej procedury możesz zapisać projekt (Save Project w oknie File) i wykonać program. Ukaże się wtedy utworzona przez Ciebie forma z oknem tekstowym i trzema przyciskami. Przyciski te możesz już naciskać i skutek naciskania (graficzny) jest na ekranie widoczny. Naciskaniu przycisków Napis i Puste nie towarzyszy jeszcze żadne działanie programu, natomiast naciśnięcie przycisku Koniec spowoduje zakończenie działania programu.



Rys. 11.4. Procedura obsługi naciśnięcia przycisku Koniec

W przypadku pozostałych dwóch przycisków wpisz tekst, który ukaże się w oknie „Komunikat". Czynisz to posługując się zmienną o nazwie KOMUNIKAT.TEXT. Zauważ, że nazwa zmiennej składa się z dwóch jakościowo różnych elementów oddzielonych kropką: pierwszy jest nazwą elementu formy, w tym wypadku okna tekstowego, drugi — rodzajem polecenia. Zmiennej tej nadajesz wartość wpisując znak równości i łańcuch objęty znakami cudzysłowu. W przypadku przycisku Puste wpisz łańcuch pusty (rys. 11.5).

Rys. 11.5. Wybór rodzaju działania obsługującego przycisk Puste

Postąp podobnie z przyciskiem Napis i użyj takiego samego polecenia, tyle że wpisz w cudzysłowach jakiś tekst.

Zauważ, że nazwy procedur, których treść wpisałeś, zawierają słowo Click, oznaczające naciśnięcie przycisku myszy („kliknięcie"). Czynności wykonywanych, które mogą być obsługiwane za pomocą procedur języka Visual Basic jest oczywiście znacznie więcej. Możesz je wybierać w oknie (rys. 11.6).

Uruchamianie programu

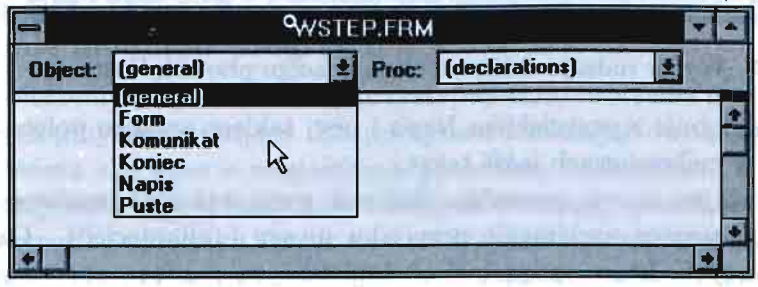
Zapisz projekt i uruchom program. Naciśnięcie przycisku Puste powinno powodować oczyszczenie zawartości okna tekstowego, naciśnięcie przycisku

Napis — wpisanie do okna tekstu, który zapisałeś w treści procedury, a naciśnięcie przycisku Koniec — zakończenie działania programu.

Otwórz projekt i zobacz, z czego się składa. Po wydaniu polecenia Open otwiera się okno projektu WSTEP. Zawiera ono nazwy elementów wchodzących w skład projektu, tj. nazwę formy FRMWSTEP i nieznane Ci nazwy plików z rozszerzeniem VBX dołączone do Twojego projektu przez system. Okno projektu zawiera dwa przyciski: View Form służący do przejścia do formy (*view* — zobacz), oraz View Code służący do obejrzenia kodu programu (rys. 11.6).

Rys. 11.6. Okno projektu WSTEP

Naciskając przycisk View Code przechodzisz do okna kodów (rys. 11.7), w którym masz nazwy przycisków sterujących i nazwę general (ogólny). Po wybraniu nazw odpowiadających przyciskom możesz obejrzeć każdą z procedur. Możesz je też zmodyfikować.



Rys. 11.7. Okno kodów projektu WSTEP

Przerwij lub zakończ pracę w systemie Visual Basic i zobacz, jakie pliki zostały utworzone przez system we wskazanym przez Ciebie katalogu. Powinny to być dwa pliki: WSTEP.MAK (plik tekstowy), zawierający wyliczenie elementów projektu, oraz WSTEP.FRM, zawierający pozostałe elementy.

Tworzenie programu wykonywalnego

Utworzony program możesz teraz skompilować poleceniem Make EXE File. Możesz tworzonemu programowi wykonalnemu nadać nazwę inną niż nazwa projektu. Jeżeli zachowasz nazwę projektu, to otrzymasz program WSTEP.EXE.

Możesz zakończyć pracę w systemie Visual Basic i uruchomić program w systemie Windows. W tym celu uaktywniasz pole menu Plik i wybierasz w nim opcję Uruchom. Możesz wpisać nazwę WSTEP.EXE, podając zarazem ścieżkę katalogu lub posłużyć się przyciskiem Przeglądaj; otworzy się wtedy okno, w którym łatwo Ci będzie odszukać plik. Możesz porównać sporządzony przez siebie program z programem załączonym na dyskietce.

Postawiłeś w ten sposób pierwsze kroki w programowaniu zorientowanym na zdarzenia. Jeżeli Cię to zainteresowało, to spytaj nauczyciela o możliwość kontynuowania w szkole nauki w tym kierunku.