11

Aplikacja. Programowanie użytkowe

Teraz, kiedy już poznałeś posługiwanie się językiem programowania wysokiego poziomu i podstawowe instrukcje umożliwiające realizowanie rozmaitych algorytmów, łatwo Ci będzie wykorzystać zdobyte umiejętności do tworzenia własnych programów i rozbudowanych makroinstrukcji w tych programach użytkowych, które takie możliwości zapewniają. Chodzi tu przede wszystkim o systemy zarządzania relacyjnymi bazami danych oraz arkusze kalkulacyjne, choć dotyczy to także niektórych edytorów tekstów.

Wiele systemów zarządzania relacyjnymi bazami danych oferuje możliwości tworzenia własnych programów za pomocą języka programowania działającego w tym systemie. Te języki, choć związane z konkretną dziedziną zastosowań, przypominają zazwyczaj język uniwersalny wysokiego poziomu. Spośród systemów omawianych w rozdz. 6 możliwości tworzenia własnych programów przez użytkownika zapewnia jedynie dBase. Jest jednak wiele innych programów przeznaczonych na komputery IBM PC, które oferują podobne możliwości; są to zarówno programy działające w systemie DOS, jak i w środowisku Windows.

Niektóre programy arkuszy kalkulacyjnych umożliwiają tworzenie makropoleceń. Makropolecenia mogą być wzajemnie powiązane między sobą, co pozwala na wykonywanie przez nie całkiem złożonych zadań. Spośród arkuszy kalkulacyjnych omawianych w rozdz. 5 możliwości takie ma QuattroPro.

Za pomocą takich własnych programów i makropoleceń można przystosować system baz danych lub arkusz kalkulacyjny do konkretnych zadań związanych najczęściej z zastosowaniami w praktyce, np. do prowadzenia rachunkowości w firmie, do kierowania wypożyczalnią, zarządzania hurtownią itp. Programy takie bywają określane mianem **aplikacji** (*application* — zastosowanie), a ich opracowywanie — **programowaniem aplikacji**.

Pojęcie aplikacji jest szeroko stosowane, obejmuje bowiem wszelkie działania przystosowujące program (pakiet) do konkretnego zastosowania, a nie tylko te, które dotyczą samych obliczeń lub przetwarzania informacji. Na przykład pojęcie aplikacji obejmuje przystosowanie wyglądu ekranu i rodzaju „przycisków" do konkretnych potrzeb, utworzenie własnego menu poleceń, a nawet pomocy do poszczególnych haseł. Niektóre pakiety użytkowe oferują użytkownikowi możliwości przystosowania wyglądu ekranu do konkretnego zadania.

Aplikacja może mieć nie tylko postać makropolecenia lub programu w języku konkretnego systemu użytkowego, ale także samodzielnego programu wykonywalnego.

11.1. Makropolecenia w arkuszu kalkulacyjnym

11.1.1. Proste makropolecenie

Przykład tworzenia prostego makropolecenia poznałeś w rozdz. 5.5.4. Dotyczył on arkusza QuattroPro, niemniej przedstawiony sposób tworzenia makropolecenia jest typowy dla różnych arkuszy kalkulacyjnych.

Ćwiczenie 11-1

Jeżeli pragniesz sobie przypomnieć, jak tworzy się makropolecenia, to powróć do przykładu z rozdz. 5.5.4. Utworzyłeś tam makropolecenie o nazwie „\Z", służące do zmiany formatu liczby w komórce bieżącej na stałopozycyjny o trzech miejscach po kropce dziesiętnej. Utwórz obecnie makropolecenie nadawania liczbie formatu wykładniczego (*scientific*). Umieść je w komórce nie przylegającej do poprzedniego makropolecenia ani z góry, ani z dołu i nazwij np. „\Y".

Zapisz teraz w arkuszu kilka liczb o długiej części ułamkowej, poszerz kolumnę(y), tak żeby były dobrze widoczne wszystkie cyfry zapisu liczb, i wykonaj kilkakrotnie na zmianę oba makropolecenia na zaznaczonym obszarze arkusza.

11.1.2. Ciągi makropoleceń

Makropolecenia mogą być bardziej rozbudowane, niemniej nie mogą być dowolnie duże, ponieważ ich wielkość jest ograniczona pojemnością komórki arkusza dla przechowywanego tekstu. Jeżeli zadanie wymaga długiego makropolecenia, nie mieszczącego się w komórce, to można użyć kilku makropoleceń. Jeżeli zapisze się je jedno pod drugim, to będą one wykonywane kolejno (od góry do dołu), podobnie jak kolejne instrukcje programu. Nie

musi się wówczas wywoływać ich oddzielnie, lecz wystarczy wywołać pierwsze z nich, a zostanie wykonany cały ich ciąg.

Przypomnij sobie, w jaki sposób tworzyłeś wykresy, wypełniając kolumnę wartościami zmiennej niezależnej zmienianej ze stałym przyrostem i obliczając dla nich wartości funkcji (rozdz. 5.3.5). Wypisz stosowane polecenia i odpowiadające im naciśnięcia klawiszy.

Gdybyś chciał zmieniać parametry funkcji lub jej postać, bądź zakres zmiennej niezależnej, to musiałbyś wszystkie polecenia powtarzać. Utworzymy zatem makropolecenia do tworzenia wykresu funkcji zawierające cały ciąg poleceń. Przy wykonywaniu kolejnych wykresów wprowadzisz jedynie nowe dane i jednorazowym naciśnięciem klawiszy wywołasz makropolecenie (bez konieczności powtarzania wszystkich poleceń oddzielnie).

Na wartości zmiennej niezależnej przeznaczymy kolumnę B od komórki B7 do B107, a na wartości funkcji — kolumnę C w zakresie tych samych wierszy.

Wzór na funkcję $F(x)$ będzie zapisywany w komórce C7, przy czym jako zmienna x traktowana będzie zawartość komórki B7. Na przykład zapiszemy w C7 wzór w postaci

```
@SIN(b7) + 0.04 * b7 * b7
```

Wartości zmiennej niezależnej wpiszemy poleceniem Fill. Dane dla tego polecenia umieścimy w komórkach B ÷ B5. W komórkach B3 i B4 podamy granice przedziału zmiennej x, np. -10 i $+10$. W komórce B5 umieścimy wzór dzielący szerokość przedziału przez 100. Obliczona setna część przedziału posłuży jako przyrost zmiennej x.

Zastanów się nad ciągiem poleceń, które umieszczą w komórkach B7 ÷ B107 wartości zmiennej niezależnej. Porównaj swój ciąg poleceń z przedstawionym:

```
/efb7..b107~b3~b5~107~
```

Zapisany ciąg poleceń oznacza, że:

- w polu menu Edit (`e` — pierwszy znak)
- poleceniem Fill (`f` — drugi znak; makropolecenie ma już dwa znaki: `ef`)
- jest objęty obszar `b7..b107` (osiem znaków i dodatkowy znak tyldy~ odpowiadający naciśnięciu klawisza *Enter*; makropolecenie ma już jedenaście znaków: `efb7..b107~`);
- przy czym w obszarze tym jako pierwsza ma być umieszczona zawartość komórki B3 (b3 i *Enter* — trzy znaki; makropolecenie ma już czternaście znaków: `efb7..b107~b3~`);

- i ma być zwiększana z krokiem równym zawartości komórki B5 (b5 i *Enter* — dwa znaki; makropolecenie ma już siedemnaście znaków: `efb7..b107~b3~b5~`)
- aż do wiersza 107 (trzy cyfry i *Enter* — ostatnie cztery znaki; makropolecenie ma już dwadzieścia jeden znaków: `efb7..b107~b3~b5~ ~107~`).

Jak pamiętasz, znaki alfabetu są zapisywane w makropoleceniach małymi literami, ponieważ podczas wydawania poleceń odpowiadające im klawisze są naciskane bez klawisza *Shift*.

Wymieniony ciąg znaków jest umieszczony jako (zwyczajny) tekst w komórce E3. Zawartość tej komórki zostanie potraktowana jako makropolecenie przez nadanie jej nazwy „\Z" (nadawanie nazwy opisano w rozdz. 5.5.4).

Przyjrzyj się teraz makropoleceniom zapisanym w komórkach E4 i E5.

Program QP-1

	A	B	C	D	E
1		Zakres	Funkcja		Wykres
2					
3	Od				/efb7..b107~b3~b5~107~
4	Do				/ecc7..c7~c8..c107~
5	Krok	0			/ggxs1c7..c107~xb7..b107~qvq

Makropolecenie zapisane w komórce E4 dotyczy powielenia wzoru zapisanego (przez użytkownika) w komórce C7 na cały ciąg komórek w tej kolumnie.

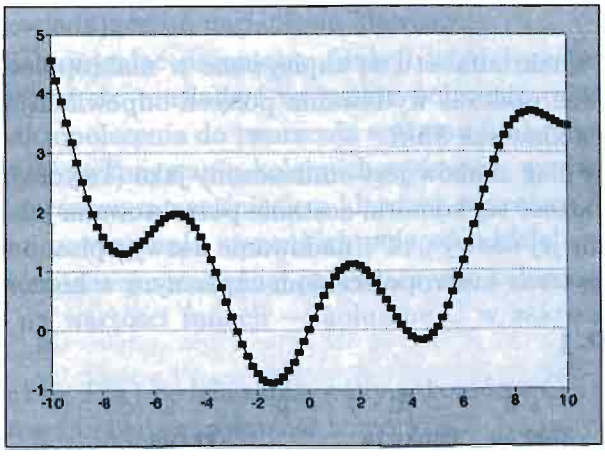
Makropolecenie w komórce E5 dotyczy tworzenia wykresu: po uaktywnieniu pola menu Graph (`g` — pierwszy znak) określany jest typ wykresu jako XY (`x` — drugi znak), potem pierwsza seria danych jako `c7..c107`, potwierdzona naciśnięciem *Enter* (`s1c7..c107~`), a następnie seria danych zmiennej niezależnej x jako `b7..b107`, również potwierdzona naciśnięciem *Enter* (`xb7..b107~`). Po wyjściu z okna poleceniem Quit zostaje wywołana funkcja pokazania wykresu View (`qv`). Po zakończeniu oglądania wykresu następuje wyjście z okna menu poleceniem Quit (do makropolecenia jako ostatni znak dochodzi litera `q`).

Makropolecenie o nazwie „\Z" uruchamiasz kombinacją klawiszy *Alt-Z*. Ponieważ bezpośrednio pod komórką E3, w której jest ono umieszczone, zapisane są inne makropolecenia, więc zostaną wykonane polecenia z komórek E3, E4 i E5.

Ćwiczenie 11-2

A. Wpisz te makropolecenia do arkusza (ewentualnie odczytaj arkusz MAK1.WQ1) i nadaj komórce E3 nazwę „\Z".

B. W komórce C7 wpisz wzór funkcji `@SIN(b7) + 0.04 * b7 * b7`, w komórce B3 wpisz -10, a w B4 wpisz $+10$ jako granice przedziału zmiennej niezależnej x, po czym wywołaj makropolecenie *(Alt-Z)*. Czy otrzymałeś wykres podobny do przedstawionego na rys. 11.1?



Rys. 11.1. Wykres funkcji $sin(x)+0.04x^2$ otrzymany za pomocą makropolecenia

C. Wpisz w komórkach B3 i B4 inne wartości granic przedziału, a w komórce C7 inną funkcję, i ponownie wywołaj makropolecenie *(Alt-Z)*. Czy otrzymałeś wykres?

Zastanów się teraz nad utworzeniem makropolecenia kasującego w arkuszu MAK1 zarówno dane do wykresu wprowadzone przez użytkownika, jak i zmienne wykresu obliczone przez arkusz. Zastanów się, gdzie byś je umieścił, żeby nie było wykonywane razem z makropoleceniem „\Z", lecz niezależnie od niego. Jeżeli wpisywałeś sam makropolecenia do arkusza, to zapisz go pod nazwą MAK1M. Porównaj opracowane przez siebie makropolecenie (ciąg poleceń) z makropoleceniami zapisanymi w komórkach E7 i E8 arkusza MAK1K. W razie wątpliwości „rozpisz" je na kolejne polecenia:

```
/gcr1~x~qqq
/eeb7..c107~/eeb3..b4~
```

Ćwiczenie 11-3

A. Posłuż się utworzonym makropoleceniem (w pliku MAK1K.WQ1 ma ono nazwę „\Y") aby skasować dane do wykresu sporządzonego makropoleceniem „\Z", wprowadź następnie nowy wzór funkcji oraz zakres zmienności jej argumentu x i ponownie sporządź wykres.

B. Umieść utworzone makropolecenie bezpośrednio pod makropoleceniem „\Z", nadaj mu nazwę i spróbuj utworzyć kolejno wykresy kilku funkcji. Który z dwóch sposobów działania (różniących się umieszczeniem makropolecenia kasującego dane) uważasz w tym zastosowaniu za wygodniejszy?

11.1.3. Język makropoleceń

Makropolecenia mogą być zapisywane także za pomocą specjalnego języka. W ten sposób zapisuje się wszystkie te polecenia, które odpowiadają wybieraniu poleceń z menu, a oprócz nich dodatkowe polecenia umożliwiające sterowanie biegiem programu i komunikowanie się z użytkownikiem.

Do najważniejszych instrukcji umożliwiających tworzenie różnorodnych algorytmów należą instrukcje warunkowe i instrukcje zmieniające kolejność wykonywania instrukcji, a w szczególności tworzenie pętli. Jest więc w języku makropoleceń instrukcja warunkowa IF; jeżeli podany po niej warunek jest spełniony, to wykonywane są wszystkie polecenia zapisane w tej samej komórce, w przeciwnym razie nie. Do zmiany kolejności służy polecenie skoku BRANCH (*odgałęzienie*), po którym podaje się nazwę makropolecenia, które ma być wykonane jako następne.

Przy wielokrotnym wykonywaniu takich samych obliczeń może być konieczne polecenie wykonania obliczeń między kolejnymi iteracjami. Służy do tego polecenie RECALC (*recalculate* — oblicz ponownie).

Instrukcje warunkowe. Skok do makropolecenia

W arkuszu QuattroPro instrukcje języka makropoleceń zapisuje się w nawiasach klamrowych. Zilustrujemy to przykładem programu wyznaczającego pierwiastek funkcji ciągłej $F(x)$, tzn. taką wartość zmiennej x, dla której wartość funkcji jest równa zeru. (Uwaga, można to zadanie wykonać także za pomocą gotowego polecenia, tu jednak chodzi o ilustrację makropoleceń).

Jako punkt wyjścia przyjmujemy, że wiesz (np. po obejrzeniu wykresu funkcji za pomocą poprzedniego makropolecenia), w jakiej okolicy funkcja $F(x)$ przekracza oś Ox i znasz taką wartość x_1, dla której funkcja ma wartość dodatnią $F(x_1) > 0$ i taką wartość x_2, dla której $F(x_2) < 0$. Gdybyś sam szukał pierwiastka, to szukałbyś go w przedziale, w którym funkcja zmienia znak. Być może nigdy nie wyznaczyłbyś go idealnie dokładnie, ale dla potrzeb praktyki wystarcza wyznaczenie rozwiązania z określoną dokładnością.

Algorytm postępowania przedstawia się następująco: W każdym kroku obliczasz wartość zmiennej x_0 leżącą pośrodku między x_1 i x_2 i odpowiadającą jej wartość funkcji. Jeżeli wartość ta jest dostatecznie bliska zera,

to kończysz postępowanie uznając, że znalazłeś rozwiązanie, i że jest nim x_0. W przeciwnym razie do dalszego postępowania wybierasz jeden z dwóch przedziałów: albo od x_1 do x_0, albo od x_0 do x_2 — ten, w którym funkcja zmienia znak. Zatem jeżeli wartość $F(x_0)$ jest dodatnia, to zmiennej x_1 przypisujesz nową wartość x_0, a jeżeli ujemna, to wartość x_0 przypisujesz zmiennej x_2. Ponieważ wartość x_0 wyznaczasz zawsze jako środek przedziału od x_1 do x_2, dzielisz więc za każdym razem przedział na połowę; z tego względu opisany sposób postępowania nosi nazwę metody bisekcji. Ponieważ za każdym razem nowy przedział jest dwukrotnie krótszy od poprzedniego, więc pierwiastek możesz wyznaczyć z dowolnie dużą dokładnością (na jaką pozwala dokładność obliczania wartości funkcji).

Obliczenia możesz wykonać w arkuszu z następującymi danymi. W komórce B3 znajduje się zmienna x_1, w B4 — x_2. Komórka B5 jest przeznaczona na x_0; znajduje się tam wzór wyznaczający wartość tej komórki jako $0,5 * (x_1 + x_2)$. W komórkach C3, C4 i C5 znajduje się skopiowany z modyfikacją adresów wzór funkcji; za każdym razem argumentem jest zawartość komórki B w tym samym wierszu. Tym samym w komórkach tych są wartości $F(x_1)$, $F(x_2)$ i $F(x_0)$.

Algorytm przedstawionego postępowania sformułowany w języku makropoleceń zawiera się w czterech komórkach D3÷ D6. W pierwszej, która ma nazwę „\Y", jest polecenie warunkowe podstawienia (*let* — niech) na zawartość komórki B3 (x_1) wartości B5 (x_0), jeżeli liczba w komórce C5 ($F(x_0)$) jest dodatnia. Podobne znaczenie ma instrukcja zawarta w komórce E4, tyle że dotyczy przypadku przeciwnego. W komórce D5 są zawarte polecenia ponowienia obliczeń w komórce B5 (nowa wartość x_0) i w obszarze C3..C5 (nowa wartość funkcji). Na końcu sprawdzany jest warunek kontynuowania obliczeń ($F(x_0)$ różni się od zera więcej niż o 0.000001); jeżeli jest spełniony, to następuje skok do pierwszego polecenia (w komórce E3).

Program QP-2

```
      A        B      C       D      E
1               x      F(x)          Makro
2
3   F(x)>0     -3       41           {if +c5>0}{let b3,b5:value}
4   F(x)<0      7      -19           {if +c5<0}{let b4,b5:value}
5   nowy        2      -14           {recalc b5..b5}{recalc c3..c5}
6                                    {if @ABS(c5)>0.000001}{branch \y}
```

Wykonaj program, zapisz go w pliku o nazwie MAK2.WQ1, a następnie spróbuj go zmodyfikować.

Makropolecenie jako procedura

Odwołanie do innego makropolecenia za pomocą instrukcji BRANCH odpowiada instrukcji skoku. W podanym przykładzie po takim odwołaniu się do makropolecenia o nazwie „\Y" znajdującego się w komórce E3 i po wykonaniu go jako następne zostaje wykonane makropolecenie z komórki E4, potem z E5 itd.

Jeżeli w odwołaniu pominiemy słowo BRANCH, pozostawiając samą nazwę makropolecenia w nawiasach klamrowych, to odwołanie nabierze charakteru odwołania do procedury. Różnica wyraża się tym, że po wykonaniu makropolecenia „\Y" sterowanie powróciłoby do komórki E6, z której nastąpiło wywołanie.

Zmodyfikujemy makropolecenie rozbijając je na dwa, przy czym dotychczasowe makropolecenie „\Y" będzie wykorzystywane jako procedura, natomiast zostanie utworzone odrębne makropolecenie, będące odpowiednikiem programu głównego zawierającego pętlę (realizowaną za pomocą skoku) i badanie warunku. (Program możesz wpisać lub odczytać z pliku MAK2P.)

Program QP-3

	A	B	C	D	E
1		x	F(x)		Makro(procedura) \Y (E3)
2					
3	F(x)>0	-3	41		{if +c5>0}{let b3,b5:value}
4	F(x)<0	7	-19		{if +c5<0}{let b4,b5:value}
5	nowy	2	-14		{recalc b5..b5}{recalc c3..c5}
6					{return}
7					
8					Makro(program) \Z (E9)
9					{\y}
10					{if @ABS(c5)>0.000001}{branch \z}

W komórce E6 została dopisana instrukcja RETURN, nadająca ciągowi instrukcji zapisanych w komórkach od E3 do E5 charakter procedury. Właściwy program główny został zapisany w komórkach E9, E10. Pierwsza z tych komórek zawiera odwołanie do makropolecenia o nazwie „\Y" jako do procedury; sama ma nadaną nazwę „\Z". Druga zawiera instrukcję skoku warunkowego do makropolecenia o nazwie „\Z" realizującą pętlę (instrukcja ta jest odpowiednikiem instrukcji z komórki E6 arkusza MAK2). Wywołanie makropolecenia o nazwie „\Z" spowoduje rozpoczęcie wykonywania obliczeń w pętli, z każdorazowym wywoływaniem procedury nazwanej „\Y". Jeżeli dane są wprowadzone prawidłowo, to po krótkim czasie obliczenia zostaną zakończone z takim samym wynikiem jak w arkuszu MAK2.

W podanym przykładzie zmiana ma charakter formalny, jedynie ilustrując tworzenie procedur. Ogólnie biorąc, możliwość odwoływania się do makropoleceń jak do procedur znacznie zwiększa możliwości tworzenia programów w języku makropoleceń.

Przykładowe instrukcje

Język makropoleceń umożliwia Ci tworzenie programów w sposób podobny do tego, jaki stosowałeś w systemie programowania QBasic lub Turbo Pascal.

- **Komunikaty** do użytkownika wypisujesz za pomocą instrukcji MESSAGE, za którą podajesz cztery parametry: 1) adres komórki zawierającej tekst komunikatu, 2) i 3) numery kolumny i wiersza ekranu, stanowiące lewy górny narożnik okna, w którym komunikat zostanie pokazany oraz 4) termin zakończenia wyświetlania komunikatu. Termin wygodnie jest określać za pomocą sumy dwóch funkcji: @NOW i @TIME; np. `@NOW+@TIME(0,0,6)` spowoduje wyświetlanie komunikatu przez 6 sekund.
- **Odczytywanie danych** wprowadzanych przez użytkownika z klawiatury wykonuje się za pomocą instrukcji GETLABEL dla tekstów i GETNUMBER dla liczb. Każda z wymienionych instrukcji ma dwa parametry: 1) tekst napisany w cudzysłowach (do 70 znaków) wyświetlany jako zachęta do wprowadzenia danych oraz 2) adres lub nazwa komórki, w której odczytane dane mają być umieszczone.
- **Odczytywanie klawiatury**; instrukcja GET powoduje czekanie na naciśnięcie jakiegokolwiek klawisza; znak odpowiadający naciśniętemu klawiszowi jest wstawiany do komórki, której adres jest parametrem instrukcji.
- **Przypisywanie wartości** komórce. Służy do tego instrukcja LET, pokazana w przykładzie, oraz instrukcja BLANK powodująca wyczyszczenie komórki o podanym adresie.
- **Pętle obliczeń** możesz tworzyć za pomocą instrukcji FOR. Ma ona pięć parametrów; są to adresy komórek zawierających: 1) licznik iteracji, 2) wartość początkową, 3) wartość końcową, 4) wartość kroku, 5) makropolecenie stanowiące wykonywaną w pętli procedurę.
- **Komentarze** tworzysz umieszczając średnik na początku tekstu objętego nawiasami klamrowymi. Taki sposób tworzenia komentarzy umożliwia łatwe przekształcenie instrukcji w komentarz (przez dodanie średnika na początku) i odwrotnie.

Są jeszcze makropolecenia umożliwiające tworzenie własnego menu i makropolecenia służące do odczytywania i zapisywania plików tekstowych.

Uwaga. Makropolecenie o nazwie „\0" (zero) jest wykonywane samoczynnie po uruchomieniu programu. Tego typu makropolecenie zamienia arkusz w specjalizowany program — aplikację.

11.1.4. Uruchamianie makropoleceń

W makropoleceniach mogą być błędy, tak jak w każdym programie. Podobnie jak w systemach programowania możesz i w języku makropoleceń korzystać z debugera. Masz do niego dostęp w menu Tools po wybraniu polecenia Macro. Debuger umożliwia wykonywanie makropoleceń w trybie pracy krokowej. Ponadto możliwe jest wstawianie „punktów kontrolnych" Breakpoints, w których wykonywanie makropolecenia zostaje zatrzymane. Można też śledzić wartości (do czterech) zmiennych w komórkach określanych jako Trace Cells.

11.1.5. Zalecenia

Przy tworzeniu makropoleceń kieruj się następującymi zasadami:

1. Poszczególne makropolecenia powinny być proste.
2. Pętle powinny być krótkie.
3. Przy długim czasie obliczeń użytkownik powinien być informowany, np. co pewien czas, żeby się nie niepokoił.
4. Należy zapewnić użytkownikowi dostatecznie dużo czasu na przeczytanie komunikatów i wyników pośrednich.
5. Makropolecenia należy gromadzić w odrębnej części arkusza (nie mieszać ich z danymi).

Ćwiczenie 11-4

Jeżeli stosowany w Twojej pracowni arkusz kalkulacyjny ma możliwość stosowania makropoleceń, to utwórz makropolecenia o charakterze programów:

A. Obliczanie kwadratu liczby wprowadzonej przez użytkownika.
B. Obliczanie rozwiązania równania kwadratowego ax^2+bx+c w dialogu z użytkownikiem (zadanie 8-1).
C. Przeprowadzanie testu z tabliczki mnożenia z zadawanymi losowo pytaniami (zadanie 8-11).

11.2. Programy w systemie baz danych dBase

System zarządzania relacyjnymi bazami danych dBase zawiera własny język programowania (rozdz. 6.4.5). Instrukcjami tego języka są przede wszystkim takie instrukcje, jakie użytkownik wydaje w formie poleceń, ale oprócz nich

są instrukcje używane wyłącznie w programach tworzonych przez użytkownika.

Program może być przygotowany pod dowolnym edytorem. Gdy jest zapisany w pliku z rozszerzeniem PRG, wówczas może być wywołany poleceniem DO z dodaną nazwą pliku (bez rozszerzenia). Można powiedzieć, że nazwa takiego pliku staje się nazwą programu.

Użytkownik ma do dyspozycji typowe elementy języka programowania wysokiego poziomu, jak zmienne numeryczne, tekstowe i logiczne, operacje arytmetyczne, operacje na tekstach i operacje logiczne, funkcje matematyczne i inne, oraz instrukcje umożliwiające sterowanie przebiegiem obliczeń: instrukcje warunkowe oraz instrukcje pętli. Znając zasady posługiwania się bazami danych i znając jakiś język programowania wysokiego poziomu, możesz z łatwością napisać program w języku dBase.

11.2.1. Przykładowy program

Przykład nawiązuje do zadania 8-8 z rozdz. 8. Rozpocznij od utworzenia bazy danych o nazwie PLIKI zawierającej nazwy plików.

Tworzenie bazy danych

Możesz założyć bazę zawierającą tylko nazwy plików (8-znakowe), ale możesz też założyć bazę mającą jeszcze inne pola. Przyjmiemy, że baza zawiera następujące pola:

1. DYSKIETKA — znakowe o szerokości 11 znaków — przeznaczone do przechowywania nazwy dyskietki, na której znajduje się plik;
2. PLIKNAZ — znakowe o szerokości 8 znaków — przeznaczone do przechowywania nazw plików;
3. ROZ — znakowe o szerokości 3 znaków — przeznaczone na rozszerzenie;
4. WIELKOSC — znakowe o szerokości 10 znaków — przeznaczone na wielkość pliku.

Przygotowanie i wprowadzenie danych

Dane do utworzonej bazy wprowadź ręcznie albo z pliku z danymi. Możesz posłużyć się jednym z plików o nazwie PLIKDIR*.TXT załączonym na dyskietce (w katalogu PLIKI\ROZNE); możesz też odpowiedni plik otrzymać, posługując się swoim programem (utworzonym w systemie QBasic lub Turbo Pascal). Jeżeli będziesz sam tworzyć plik z danymi przy użyciu edytora, to skopiuj pewną ilość wierszy kilkakrotnie, żeby niektóre nazwy plików wystąpiły więcej niż jeden raz.

Dane do Twojej bazy wczytaj poleceniem `APPEND FROM plikdirn DELIMITED` (uwaga, nazwę pliku podajesz bez apostrofów i bez rozszerzenia TXT). Wykonaj to polecenie w trybie z kropką.

Sortowanie danych

Obejrzyj utworzoną bazę PLIKI poleceniem `BROWSE`. Jeżeli posłużyłeś się odpowiednio dużym plikiem, np. PLIKDIR4.TXT, to widzisz, że wyszukanie ręczne plików o identycznych nazwach jest niemal niemożliwe.

Uporządkuj bazę danych sortując ją. Wydaj w tym celu polecenie

```
SORT TO pliki2 ON pliknaz
```

Spowoduje ono utworzenie bazy danych o nazwie PLIKI2, mającej te same rekordy co baza PLIKI, ale umieszczone zgodnie z nowym porządkiem, określonym przez nazwy pól wymienione po słowie ON. Mógłbyś wymienić jeszcze inne pola, oddzielając ich nazwy przecinkami, np. PLIKNAZ, WIELKOSC, ROZ. Wtedy rekordy uporządkowane według nazwy, w obrębie rekordów o takich samych nazwach zostałyby dodatkowo uporządkowane według wielkości itd.

Otwórz utworzoną bazę poleceniem `USE PLIKI2` i obejrzyj ją poleceniem `BROWSE`. Zapewne teraz łatwiej Ci wyłowić powtarzające się nazwy plików. Jeżeli jednak posługujesz się plikiem zawierającym kilkaset lub więcej rekordów, to takie ręczne ich wyszukiwanie staje się nużące; co więcej łatwiej wtedy o popełnienie omyłek. Spróbujemy teraz tę czynność wykonać za pomocą programu użytkownika. Zakończ pracę z systemem dBase poleceniem `QUIT`.

Pierwszy program

Na początek proste ćwiczenie dla nabrania wprawy. Przygotuj pod edytorem plik ASCII z kilkoma poleceniami, którymi się posługiwałeś. Na przykład napisz w nim:

Program DB-1

```
USE pliki2
BROWSE
```

Zapisz go pod nazwą DB1.PRG. Teraz uruchom system dBase (w trybie z kropką) i jako pierwsze wykonaj polecenie `DO DB1`. Dla systemu dBase oznacza to, że ma otworzyć plik DB1.PRG i wykonywać kolejno zawarte w nim polecenia. W tym wypadku system powinien otworzyć bazę PLIKI2 i umożliwić Ci oglądanie jej w trybie tabelarycznym (`BROWSE`).

Program oczywiście może być bardziej rozbudowany.

Program szukania takich samych nazw

Zaczniemy od znalezienia liczby rekordów w bazie o takiej samej nazwie jak podana przez użytkownika. Założymy, że baza danych znajduje się w pamięci programu i jest otwarta. Zastanów się, jakiego rodzaju instrukcji potrzebujesz do napisania programu dla algorytmu rozwiązywania zadania 8-15. Powinieneś umieć:

- odczytać nazwę podaną przez użytkownika i zapisać ją w pamięci;
- odczytać pole PLIKNAZ;
- zacząć odczytywanie pola PLIKNAZ od pierwszego rekordu;
- podjąć działanie zależnie od wyniku porównania nazwy odczytanej z zapamiętaną;
- nadać nową wartość zmiennej określającej liczbę rekordów o danej nazwie;
- przesuwać się w tablicy do kolejnych rekordów;
- rozpoznać koniec tablicy z rekordami i zakończyć działanie;

Przyjrzyj się programowi DB-2.

Program DB-2

```
SET TALK OFF
GO TOP
ACCEPT 'Podaj nazwe pliku: ' TO nazwapliku
STORE 0 TO liczba
DO WHILE .NOT. eof()
  IF lower(pliknaz) = lower(nazwapliku)
     ? nazwapliku
     STORE liczba + 1 TO liczba
  ENDIF
  SKIP
ENDDO
IF liczba > 1
  ? 'Plik ', nazwapliku, ' wystepuje ', liczba, ' razy'
ELSE
  ? 'Nie ma takiego pliku'
ENDIF
SET TALK ON
```

Instrukcje początkowa i końcowa służą do tego, żeby instrukcje programu nie ukazywały się na ekranie podczas wykonywania go (*set talk off* — wyłącz rozmawianie; *on* — włącz). Instrukcja GO TOP (idź do góry) służy do przemieszczenia wskaźnika wiersza bieżącego na początek tablicy. Do przemieszczania wskaźnika o jedno miejsce w dół służy instrukcja SKIP (*przeskocz*). Instrukcja ACCEPT TO *nazwapliku* jest przeznaczona do nada-

nia zmiennej *nazwapliku* wartości odczytanej z klawiatury, przy czym napis umieszczony w apostrofach między słowami ACCEPT i TO jest drukowany jako zachęta do wprowadzenia danych (*accept* — przyjmij). Instrukcja STORE *wartość* TO *zmienna* nadaje zmiennej wartość zapisaną między słowami STORE i TO (*store* — przechowaj). Instrukcja rozpoczynana znakiem zapytania powoduje drukowanie napisów i wartości zmiennych na ekranie (odpowiednik instrukcji print).

Rozpoznajesz zapewne w programie instrukcje warunkowe, zaczynające się słowem IF i kończące słowem ENDIF, z których jedna zawiera także słowo ELSE i umożliwia podjęcie działań w razie niespełnienia warunku. Domyślasz się też zapewne, że konstrukcja DO WHILE ENDDO tworzy pętlę, w której obliczenia są kontynuowane dopóki spełniony jest warunek za słowem WHILE. Rozpoznajesz też słowo NOT, zauważając, że pisane jest z kropkami po bokach. Z symbolem EOF miałeś już do czynienia w przypadku plików; tu odnosi się on do końca tabeli baz danych; gdy odczytywany jest ostatni wiersz, wówczas funkcja EOF przyjmuje wartość Prawda. Warunek porównania nazw plików jest zapisany z użyciem funkcji LOWER (*niższy*), zamieniającej wielkie litery na małe; podobne funkcje poznałeś w systemie QBasic lub Turbo Pascal. W nazwach plików systemu DOS małe i wielkie litery nie są rozróżniane, korzystne jest więc zastosowanie takiej funkcji, żeby uniknąć nieporozumień wynikających ze stosowania wielkich bądź małych liter.

Program odczytuje nazwę podaną przez użytkownika. Następnie odczytuje pole PLIKNAZ otwartej bazy danych od pierwszego wiersza do ostatniego, porównując z nazwą podaną przez użytkownika; jeżeli są zgodne (z dokładnością do wielkości liter), to nazwa jest drukowana na ekranie i jednocześnie jest zwiększana o jeden wartość zmiennej *liczba*. Po dojściu do końca tabeli jest drukowany komunikat końcowy.

Ćwiczenie 11-5

Sprawdź działanie programu na Twoich bazach danych. Uruchom system i otwórz bazę, po czym wydaj polecenie DO db2. Czynności te powtórz kilkakrotnie dla różnych nazw plików i ewentualnie dla różnych baz.

Program szukania plików występujących wielokrotnie

Program nie zawiera nowych instrukcji. Jest zrealizowany według algorytmu 19 (choć nie w każdym szczególe) opracowanego dla bazy danych posortowanej według nazw. Pisząc program przyjmujemy ponadto, że baza danych jest już otwarta.

Zmienna *liczbaplw* służy do zliczania plików, których nazwy występują więcej niż jeden raz. Nazwa bieżącego pliku jest przechowywana w zmiennej *nazwabiez*. Do zliczania, ile razy występuje plik o takiej samej nazwie jak *nazwabiez*, służy zmienna *liczba*. Program działa w ten sposób, że wskaźnik wiersza bieżącego jest przesuwany tak długo, jak długo nazwa pliku jest zgodna z nazwą bieżącą. Jeżeli zostaje znaleziony plik o innej nazwie, to drukowany jest komunikat dotyczący plików o danej nazwie, a program podejmuje działanie od tego rekordu, na którym poszukiwanie identycznych nazw zostało przerwane. W ten sposób cała baza danych jest przeszukiwana tylko jeden raz.

Program DB-3

```
SET TALK OFF
GO TOP
STORE 0 TO liczbaplw
DO WHILE .NOT. eof()
  STORE lower(pliknaz) TO nazwabiez
  SKIP
  STORE 1 TO liczba
  DO WHILE (.NOT. eof()) .AND. (lower(pliknaz) = nazwabiez)
       STORE liczba + 1 TO liczba
       SKIP
  ENDDO
  IF liczba > 1
     ? 'Plik ', nazwabiez, ' wystepuje ', liczba, '-krotnie'
     STORE liczbaplw + 1 TO liczbaplw
  ENDIF
ENDDO
  ?
  ? 'W bazie danych ', liczbaplw, ' nazw plikow wystepuje
                                                  wielokrotnie'
SET TALK ON
```

Ćwiczenie 11-6

Sprawdź działanie programu DB-3 na Twoich plikach. Możesz dopisać do bazy danych pliki o takich samych nazwach i sprawdzić ponownie.

11.2.2. Elementy języka programowania w systemie dBase

Język programowania należący do systemu zarządzania bazami danych dBase jest bardzo rozbudowany.

Przede wszystkim umożliwia **operowanie wieloma bazami danych** (tabelami). Służy do tego instrukcja SELECT. Na początku programu

otwiera się potrzebne bazy danych za pomocą ciągu instrukcji w rodzaju:

```
SELECT 1
USE pliki
SELECT 2
USE pliki3
```

po czym w programie uaktywnia się właściwą bazę danych instrukcją SELECT z nazwą pliku (lub numerem). Od tego miejsca do następnej instrukcji SELECT wszystkie instrukcje odnoszą się do aktywnej bazy.

Do **sterowania** przebiegiem programu służą dodatkowo instrukcje wyboru DO CASE i ENDCASE — para instrukcji podobnych do stosowanych w uniwersalnych językach programowania. Są też instrukcje umożliwiające przerywanie obliczeń w pętli DO WHILE: LOOP (skrócenie pętli z powrotem do badania warunku) i EXIT (zakończenie całych obliczeń w pętli).

Do dialogu z użytkownikiem umiesz już stosować instrukcje ACCEPT i „?". ACCEPT służy do wprowadzania danych tekstowych, zaś do wprowadzania liczb, a nawet wartości wyrażeń, służy instrukcja INPUT. Podobną funkcję ma instrukcja WAIT (*czekaj*), wstrzymująca działanie programu do chwili naciśnięcia przez użytkownika jakiegoś klawisza i przypisująca wartość tego klawisza zmiennej.

Do przedstawiania długich tekstów służy instrukcja TEXT, ENDTEXT.

Duże znaczenie ma możliwość tworzenia formularzy i przedstawiania w tych formularzach wyników oraz odczytywania danych wprowadzanych przez użytkownika. Służy do tego rodzina instrukcji zaczynających się od symbolu `@`. Polecenia te wyrażają się we współrzędnych ekranu tekstowego: 25 wierszy i 80 kolumn; współrzędne 0,0 odnoszą się do lewego górnego rogu.

```
@1,0 to 23,79
@4,30 SAY 'FORMULARZ WYNIKOW'
@8,10 SAY 'Nazwa pliku: ' GET nazwapli
```

Pierwsze polecenie rysuje ramkę; jeżeli umieścić by za nim słowo DOUBLE, to ramka byłaby narysowana podwójną linią. Drugie umieszcza napis począwszy od wskazanego miejsca na ekranie. Trzecie umieszcza napis i umożliwia wprowadzenie danych.

Możliwe jest ponadto tworzenie procedur.

Ćwiczenie 11-7

1. Zmodyfikuj program DB-2 w taki sposób, żeby dane były odczytywane z formularza.
2. Zmodyfikuj program DB-3 tak, żeby wyniki końcowe były przedstawiane w formularzu.

11.2.3. Podsumowanie

Zastanów się nad znanymi Ci bazami danych i pomyśl o pytaniach, jakie można by stawiać, a które wykraczają poza typowe zapytania.

Rozważ na przykład bazę danych zawierającą informację o plikach zapisanych na poszczególnych dyskietkach i zastanów się, jakich informacji mógłbyś potrzebować i dla których z nich trzeba byłoby napisać program. Niech ta baza danych ma strukturę podobną do bazy danych PLIKI, będącej przedmiotem działań w tym rozdziale, z tym że mogłaby zawierać dodatkowe pola, np. daty i godziny.

Raporty z zawartością poszczególnych dyskietek można wykonać za pomocą zwykłych poleceń. Wyobraź sobie jednak sytuację, w której chciałbyś odzyskać trochę miejsca na swoich dyskietkach. Na początek mógłbyś chcieć znaleźć pliki, które występują więcej niż dwa razy. Mógłbyś przy tym zastosować dodatkowe kryteria wyboru, żeby program wskazywał pliki starsze jako pliki do usunięcia. Mógłbyś też stawiać pytanie typu: które z plików zapisanych na danej dyskietce mogę skasować (ponieważ na innych dyskietkach są te same ich kopie), a które muszę skopiować na inne przed użyciem dyskietki do innego celu. Mógłbyś wreszcie za pomocą odpowiedniego programu wyszukać taką dyskietkę, którą można zastosować do innego celu bez konieczności kopiowania zawartych w niej plików.

Spróbuj teraz w podobny sposób zastanowić się nad bazą danych służącą np. do obsługi biblioteki szkolnej lub pomocy w organizacji egzaminów wstępnych do szkoły.

11.2.4. Język zapytań SQL

W niektórych systemach zarządzania relacyjnymi bazami danych można posługiwać się językiem zapytań SQL (*Structured Query Language* — strukturalny język zapytań). Ma on ten walor, że bywa stosowany na wielu różnych komputerach działających pod rozmaitymi systemami operacyjnymi. Wiele systemów zarządzania bazami danych pozwala na formułowanie zapytań w tym języku niezależnie od ich własnego języka.

Język SQL może być stosowany w trybie interaktywnym oraz do pisania programów. Jest on bardziej wydajny niż język stosowany w systemie dBase, tzn. uzyskuje się podobny rezultat przy użyciu mniejszej liczby instrukcji.

Podstawową strukturą bazy danych jest tablica. Wiersz tablicy odpowiada rekordowi. Nazwy kolumn tablicy, odpowiadające nazwom pól, zwane są atrybutami. Typy danych określa się nazwą i niekiedy także parametrem oznaczającym długość, np. INTEGER — zmienna całkowita; DECIMAL(n,m) — liczba rzeczywista w zapisie stałopozycyjnym mającym n cyfr

łącznie z kropką dziesiętną, z tego m cyfr po kropce; CHAR(n) — ciąg n znaków.

Tablicę tworzy się za pomocą jednej instrukcji. Na przykład tablicę BAZA1SQ odpowiadającą rozważanej w rozdz. 6.2.2 bazie książek o nazwie BAZA1 sporządza się następująco:

```
CREATE TABLE baza1sq
 ( Nazwisko CHAR(10),
   Imie     CHAR(10),
   Tytul    CHAR(30),
   Pozyczyl CHAR(10))
```

Za pomocą instrukcji INSERT można do utworzonej tablicy wprowadzać dane.

```
INSERT INTO baza1sq
 ( Nazwisko, Imie, Tytul, Pozyczyl)
VALUES ('Sienkiewicz', 'Henryk', 'Quo vadis', 'Maciek')
```

Jedną z najważniejszych instrukcji języka SQL jest SELECT, służąca do odczytywania zawartych w bazie informacji. Odpowiada ona formułowaniu zapytań. Jej najprostsza wersja ma postać:

```
SELECT nazwy kolumn
  FROM nazwy tablic
  WHERE warunek
```

Na przykład instrukcja sporządzenia listy tytułów książek wypożyczonych przez Maćka może mieć postać:

```
SELECT Tytul
  FROM baza1sq
  WHERE Pozyczyl = 'Maciek'
```

Gdyby baza danych BAZA1SQ zawierała te same dane, co BAZA1 po odczytaniu danych z dyskietki, to na liście oprócz „Quo vadis" znalazłaby się jeszcze książka: „Sztuka przyjaźni" (porównaj z przykładem z rozdz. 6.3.10).

Zauważ, że instrukcja SELECT umożliwia przedstawienie wybranych pól (niekoniecznie wszystkich pól rekordu), realizuje więc te funkcje, które przy omawianiu baz danych były określane mianem zapytań lub filtrów (rozdz. 6.3.11).

Warunek formułowany za słowem WHERE może zawierać wyrażenia logiczne tworzone z użyciem operatorów logicznych: AND, OR i NOT, na przykład

```
  WHERE Nazwisko = 'Mickiewicz' AND Pozyczyl >= 'L'
```

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.