9

QBasic — elementy programowania

9.1. Podstawowe informacje o języku Basic

Język programowania Basic został opracowany w latach sześćdziesiątych, żeby ułatwić studentom naukę programowania. Słowo *basic* znaczy *podstawowy*; sama nazwa jest skrótem od „Beginners All-purpose Symbolic Instruction Code", co w wolnym tłumaczeniu oznacza: „wszechstronny symboliczny język programowania dla początkujących".

Różne firmy programistyczne tworzyły programy tłumaczące dla języka Basic, np. BasicA, GW-Basic (mówimy w takim przypadku o implementacjach języka Basic, pomimo że język ten nie jest zdefiniowany jednym ogólnie przyjętym standardem). System programowania (program) QBasic został opracowany przez firmę Microsoft; jest on dołączany do systemu operacyjnego MS DOS począwszy od wersji 5.0.

System QBasic wywołujesz poleceniem `QBASIC`. Możesz do tego polecenia dołączyć od razu nazwę pliku, np. MOJPROG.BAS, przy czym w przypadku rozszerzenia BAS nie musisz go podawać. Tak więc w odpowiedzi na polecenie `QBASIC MOJPROG` system szuka pliku MOJPROG.BAS w katalogu bieżącym. Jeżeli plik taki będzie znaleziony, to zostanie otwarty i jego zawartość ukaże się na ekranie, jeżeli zaś pliku takiego nie ma, to zostanie utworzony.

Do edycji programów źródłowych w systemie QBasic jest stosowany ten sam edytor Edit, którym posługiwałeś się już do edycji plików ASCII, znasz więc już jego podstawowe polecenia. W systemie QBasic edytor ten ma pewne dodatkowe możliwości.

Programy będziemy oznaczać symbolem QB z numerem oddzielonym kreską. Dla plików programów źródłowych będziemy używać tych samych

nazw, tyle że z numerem dwucyfrowym (ze względu na sposób porządkowania plików na dysku) i bez kreski oddzielającej, np. programowi QB-7 odpowiadać będzie plik QB07.BAS.

9.2. Pierwsze kroki w systemie QBasic

Zanim przystąpisz do systematycznego zapoznawania się z elementami programów i ze sposobem ich stosowania, przyjrzyj się czterem niewielkim programom i zobacz na ich przykładzie, jak wygląda edycja i uruchamianie programu i z jakimi problemami możesz się spotkać. Zarazem przekonaj się, że realizowanie w programie różnych elementów algorytmów jest dość proste.

9.2.1. Komunikat na ekranie

Pierwszy program zawiera często spotykany element wszystkich programów, jakim jest komunikat. Wywołaj system QBasic, rozpoczynając zarazem edycję pliku QB01.BAS (poleceniem `QBASIC QB01`). Następnie napisz podany tekst programu (tekst w cudzysłowach możesz zastąpić innym). Każdą linię tekstu kończ naciśnięciem klawisza *Enter*.

```
cls
print "Pierwszy komunikat: koniec programu."
```

Zauważyłeś, że po zakończeniu edycji kolejnych linii tekstu programu małe litery w słowach `cls` oraz `print` zamieniane są na ekranie na litery wielkie `CLS` i `PRINT`. W rezultacie Twój pierwszy program źródłowy ma postać:

Program QB-1

```
CLS
PRINT "Pierwszy komunikat: koniec programu."
```

Zaobserwowana zmiana świadczy o tym, że system QBasic dokonał wstępnego sprawdzenia poszczególnych linii programu źródłowego (jak pamiętasz jest to interpreter tłumaczący program źródłowy linia po linii i wykonujący od razu odpowiedni fragment programu wynikowego), zidentyfikował słowa: `cls`, `print` jako mające dla niego specjalne znaczenie i zamienił w nich małe litery na wielkie.

Dla wyjaśnienia: CLS (*CLear Screen*) jest nazwą instrukcji wyczyszczenia ekranu, zaś PRINT (drukuj) jest nazwą instrukcji wyprowadzania wyniku na ekran lub inne urządzenie, np. na drukarkę albo do pliku dyskowego. Na prawo od słowa PRINT umieszcza się to, co ma zostać wy-

prowadzone; w naszym przykładzie jest to tekst komunikatu umieszczony pomiędzy znakami cudzysłowu.

Od tej pory będziemy się stosować w książce do konwencji systemu QBasic pisania wielkimi literami nazw instrukcji i innych słów wyróżnionych przez system. System dokonuje także innych zmian, np. dodając odstępy pomiędzy składnikami instrukcji.

Uwaga. Działania wykonywane przez instrukcje CLS i PRINT odpowiadają procedurom (wbudowanym). Ponieważ w systemie QBasic procedury wbudowane nie są odróżniane od instrukcji, to i my będziemy w tym rozdziale używać jednego określenia — instrukcja.

Zapisz utworzony program źródłowy (polecenie Save w menu File, które uaktywniasz naciśnięciem klawisza *Alt*).

Następnie zainicjuj tłumaczenie i wykonywanie Twojego programu źródłowego poleceniem Start wybranym z menu Run; uzyskasz to samo naciskając klawisze *Shift-F5* bez uaktywniania menu.

Powinien zniknąć ekran edycyjny z tekstem programu źródłowego i ukazać się **ekran roboczy**, służący do wyprowadzania wyników wykonywanego programu. W górnej jego części powinien być umieszczony tekst Twojego komunikatu:

```
Pierwszy komunikat:  koniec programu.
```

jako efekt działania instrukcji PRINT, zaś w dolnej części ekranu — napis będący komunikatem systemu QBasic do Ciebie:

```
Press any key to continue
```

oznaczający

W celu kontynuacji naciśnij dowolny klawisz.

Po naciśnięciu dowolnego klawisza znika ekran roboczy i powracasz do ekranu edytora z Twoim programem źródłowym QB01 na ekranie. Ekran roboczy programu z wynikami działania możesz obejrzeć ponownie naciskając klawisz *F4*.

Jeżeli chcesz wyjść z systemu QBasic, postąp tak samo jak w edytorze Edit (polecenie Exit w menu File). Uczyń to obecnie.

Sprawdź po zakończeniu pracy w systemie QBasic, że w katalogu bieżącym znajduje się świeżo utworzony plik QB01.BAS. Odczytaj jego zawartość za pomocą programu niezależnego od systemu QBasic, na przykład poleceniem `TYPE QB01.BAS`. Zwróć uwagę, że tekst programu zapisany w pliku zawiera zmiany wprowadzone przez system QBasic (wielkie litery w nazwach instrukcji).

Ćwiczenie 9-1

Zmodyfikuj tekst programu QB-1 wprowadzając kolejno zaproponowane zmiany:
A. Usuń instrukcję CLS.
B. Zastąp tekst w cudzysłowach innym.
Uruchom program i zobacz, jaki jest skutek wprowadzonych zmian. Zmodyfikowanych plików nie zapisuj (lub zapisz pod inną nazwą, na przykład QB01ABC).

9.2.2. Czytanie danych

Drugi program zawiera nowe elementy programów w postaci instrukcji czytania danych, oraz zmiennych, którym są nadawane odczytane wartości. Skopiuj do katalogu bieżącego plik QB02.BAS z załączonej dyskietki (znajdziesz go w katalogu PROGRAMY\QBASIC). Uruchom system QBasic, wskazując nazwę pliku QB02.BAS. Możesz także nie kopiować pliku i tekst drugiego programu wpisać samemu.

Program QB-2

```
REM program QB-2  - czytanie danych
REM "Elementy informatyki" G. Ploszajski
CLS
PRINT "Jak masz na imie? (Napisz swoje imie i nacisnij Enter)"
INPUT imie$
PRINT "Hej, "; imie$, "Ile masz lat?"
INPUT wiek
PRINT wiek; "lat to piekny wiek"
```

Dla wyjaśnienia: REM (*remark* — uwaga) oznacza komentarz, podobnie jak w plikach typu *.BAT (rozdz. 2.5.1). Cały tekst umieszczony w linii za słowem REM jest przez system QBasic pomijany przy tłumaczeniu i wykonywaniu programu; jest on przeznaczony dla osób oglądających tekst programu. W plikach programów zamieszczonych na dyskietce są umieszczone komentarze (zgodnie z zaleceniem z rozdz. 8.5.1). Przytaczając w książce teksty programów komentarze te będziemy z reguły pomijać ze względu na szczupłość miejsca.

INPUT (wejście, wprowadzane dane) jest nazwą instrukcji wprowadzania danych. Wprowadzone dane są umieszczane w zmiennych *imie$* oraz *wiek*, zapisanych za słowem INPUT; zmienna *imie$* jest przeznaczona na tekst, a zmienna *wiek* na liczbę. W instrukcjach PRINT jest po kilka elementów oddzielonych średnikiem lub przecinkiem: teksty umieszczone w cudzysłowach oraz nazwy zmiennych.

Jeżeli wpisywałeś program ręcznie, to zapisz go na dysku. Uruchom go. W górnej części ekranu roboczego powinien być umieszczony napis:

```
Jak masz na imie?  (Napisz swoje imie i nacisnij Enter)
```

jako efekt działania instrukcji PRINT, a poniżej — znak zapytania jako efekt działania instrukcji INPUT, „zapraszającej" Ciebie do wprowadzenia danych. Gdy napiszesz swoje imię (lub inny tekst), kończąc naciśnięciem klawisza *Enter*, to w następnej linii powinien pojawić się napis `Hej`, razem z podanym przez Ciebie imieniem i pytanie o Twój wiek. Gdy podasz, ile masz lat, wówczas program napisze odpowiedź zawierającą na początku podaną przez Ciebie liczbę, a w dolnej części ekranu wyświetli komunikat, żebyś nacisnął jakiś klawisz.

Jeżeli popełnisz błąd, to system może odpowiedzieć komunikatem:

```
Redo from start
```

oznaczającym:

powtórz operację od początku.

Ćwiczenie 9-2

Uruchom kilkakrotnie program QB-2 sprawdzając, jaki będzie efekt:

A. Wprowadzenia liczby zamiast imienia.
B. Wprowadzenia imienia zamiast liczby.
C. Niewprowadzenia niczego (samego naciśnięcia klawisza *Enter*).
D. Wprowadzenia liczby ułamkowej z kropką dziesiętną.
E. Wprowadzenia liczby ułamkowej z przecinkiem dziesiętnym.

Czy zaobserwowałeś, jak się przejawia skutek użycia w instrukcji PRINT przecinka zamiast średnika?

9.2.3. Obliczenia

W instrukcji PRINT może wystąpić nie tylko sama zmienna liczbowa, lecz także wyrażenie arytmetyczne. Wyrażenia arytmetyczne zapisujesz używając podobnych symboli jak w arkuszu kalkulacyjnym, tzn. znaku gwiazdki * jako symbolu mnożenia i znaku kreski ułamkowej jako symbolu dzielenia; zakres oraz kolejność działań określasz za pomocą nawiasów.

Wzorując się na poznanych programach, możesz napisać program realizujący algorytm 1. Wielkościom oznaczonym w algorytmie symbolami a, b oraz h możesz przypisać zmienne o takich samych nazwach lub o nazwach, które będą wyrażały znaczenie tych wielkości. Nie powinieneś mieć trudności z utworzeniem programu, pomijając ewentualne wątpliwości co do zasad nadawania nazw zmiennym (np. czy nazwy mogą mieć dwanaście znaków).

Program QB-3

```
REM program QB-3  obliczanie zuzycia paliwa (algorytm 1)
CLS
PRINT "Podaj aktualny stan licznika kilometrow  (nacisnij
                                                    Enter)"
INPUT StanAktualny
PRINT "Podaj poprzedni stan licznika kilometrow  (nacisnij
                                                    Enter)"
INPUT StanPoprzedni
PRINT "Podaj zuzycie paliwa w litrach  (nacisnij Enter)"
INPUT Zuzycie
PRINT "Srednie spalanie wynosi ";
PRINT 100 * Zuzycie/(StanAktualny - StanPoprzedni); "l/100 km"
```

Uwaga. Każdą instrukcję zapisuj w jednej linii, bez przenoszenia do następnej. W książce zastosowano przenoszenia ze względu na brak miejsca.

W programach, które miałyby realizować następne algorytmy, potrzebujesz instrukcji umożliwiających badanie warunku i wykonywanie jednych instrukcji lub innych zależnie od tego, czy warunek jest spełniony, czy nie, obliczanie pierwiastka kwadratowego oraz wielokrotne powtarzanie tych samych obliczeń w pętli. Poza tym powinieneś wiedzieć, jakimi zmiennymi możesz operować, jak je nazywać i jak nadawać im wartości. Do pisania i uruchamiania programów przydadzą Ci się wiadomości o korzystaniu z edytora systemu QBasic oraz o pomocy, z jakiej ewentualnie mógłbyś korzystać.

9.3. Edycja i uruchamianie programów źródłowych w systemie QBasic

9.3.1. Rozpoczynanie i kończenie edycji

Edytor stosowany w systemie QBasic umożliwia wykonywanie wszystkich podstawowych operacji edycyjnych; większość z nich pokrywa się z operacjami edytora Edit, kiedy jest on wywoływany poleceniem EDIT.

Pracę z nowym plikiem możesz rozpocząć z poziomu systemu operacyjnego, wywołując system poleceniem QBASIC wraz z nazwą pliku, jak to uczyniłeś w przypadku programu źródłowego QB02. System bierze wtedy pod uwagę pliki z rozszerzeniem BAS (czego nie czyni edytor wywołany poleceniem EDIT).

Możesz wywołać system QBasic w taki sposób, żeby wskazany przez Ciebie plik programu został uruchomiony zanim będzie pokazany na ekranie

edycyjnym. Powinieneś w tym celu dodać parametr `/RUN` przed nazwą pliku, np. `QBASIC /RUN QB02`.

Jeżeli wywołasz system (program) QBasic nie podając nazwy pliku, to żeby rozpocząć pracę z edytorem powinieneś nacisnąć klawisz *Esc* (naciśnięcie klawisza *Enter* spowoduje wejście do pomocniczego programu pomocy, z którego możesz wyjść za pomocą klawisza *Esc*). Plikowi zostaje nadana tymczasowa nazwa Untitled (nie nazwany). Swoją nazwę nadasz temu plikowi w chwili zapisywania go na dysk poleceniem Save w menu File (uaktywnianym naciśnięciem klawisza *Alt* albo za pomocą myszy). Jeżeli nie określisz rozszerzenia, to system doda standardowe rozszerzenie BAS.

Dobrze jest nadawać plikom z tekstami programów nazwy, które będą Ci się kojarzyły z treścią programu lub z jego przeznaczeniem. W książce ze względu na dużą liczbę programów posługujemy się oznaczeniem liczbowym. Możesz jednak nazwy te uzupełnić, jeżeli dzięki temu będzie Ci łatwiej korzystać z programów, np. zmienić nazwę pliku QB02.BAS na QB02IMIE.BAS.

Zapamiętaj!
Nadawaj programom nazwy sygnalizujące ich treść.

Do rozpoczęcia edycji nowego pliku w trakcie działania systemu QBasic powinieneś wydać polecenie New z menu File (nowy); plik otrzymuje nazwę Untitled, którą powinieneś zmienić podczas zapisywania go na dysk.

Do kontynuowania pracy z plikiem istniejącym służy polecenie Open wybierane z menu; po wydaniu tego polecenia system otwiera okno, w którym możesz wpisać nazwę pliku lub wybrać spośród plików pokazanych na ekranie. Do przechodzenia między poszczególnymi polami służy klawisz *Tab*, a do zatwierdzania wyboru pliku z listy klawisz *Enter*; możesz także posługiwać się myszą.

Do zapisania pliku na dysku pod jego własną nazwą służy polecenie Save wybierane z menu File. Do zakończenia edycji służy polecenie Exit (*Alt-F* i *X*). Jeżeli plik nie jest przedtem zapisany na dysku, to system wyświetla komunikat:

```
Loaded file is not saved.  Save it now?
```

oznaczający:

Załadowany (do edytora) plik nie jest zapisany. Czy zapisać go teraz?

Możesz wybrać klawiszem *Tab* jedną z opcji: OK (tak), No (nie), Cancel (anuluj (to polecenie)) i Help (pomoc) lub też użyć w tym celu klawiszy z wyróżnionymi literami.

9.3.2. Wstępna kontrola poprawności programu podczas edycji

System QBasic wspomaga edycję programu źródłowego. Każda linia programu (deklaracja, instrukcja, komentarz) jest kontrolowana i w razie znalezienia błędów, system interweniuje po zakończeniu jej edycji (tj. po naciśnięciu klawisza *Enter*, a nawet po przejściu kursorem do innej linii). Jedna z form interwencji polega na poprawianiu zapisu, inna na zgłaszaniu zastrzeżeń (przy czym komunikaty są podawane po angielsku).

Niektóre słowa system rozpoznaje jako „swoje", jak to uczynił zmieniając pisownię instrukcji CLS i PRINT z małych liter na wielkie. Ogólnie słowa takie są nazywane **słowami kluczowymi** i obejmują instrukcje i deklaracje. Inne zapisane przez Ciebie teksty system analizuje w kontekście rozpoznanych słów kluczowych (lub ich braku).

Poprawianie usterek

Niektóre usterki zapisu system poprawia sam, jak gdyby uznając, że intencje autora programu są oczywiste (co nie musi być prawdą — patrz punkt C w następującym ćwiczeniu).

Ćwiczenie 9-3

A. Wywołaj system QBasic wskazując mu plik `QB01` z Twoim pierwszym programem. W drugiej linii programu (instrukcja PRINT) skasuj drugi cudzysłów i naciśnij klawisz *Enter*. Sprawdź, że system sam doda ten cudzysłów.

B. Zapisz w instrukcji PRINT kilka elementów nie oddzielając ich od siebie spacjami, na przykład `PRINT x;y;x+y`, i sprawdź, że system doda spacje.

C. Pamiętasz, że nie powinieneś stosować przecinka dziesiętnego w zapisie liczby i że przecinek jest stosowany do oddzielania różnych elementów w tej samej instrukcji PRINT, niemniej „pomyl się" teraz i dopisz w trzeciej linii instrukcję drukowania dwóch liczb w postaci: `PRINT 22.34 56,78`. Zobacz, czy system zmodyfikuje tak zapisane liczby i jak. Czy wiesz, jak te liczby będą wyświetlane na ekranie? Wykonaj program i sprawdź.

Wskazywanie błędów

Są błędy, które system jedynie sygnalizuje, „nie wiedząc" jak je poprawić albo uznając, że powinien to zrobić autor.

Na przykład dopisz słowo PRINT za instrukcją CLS i zakończ tę linię programu. System nie akceptuje tak zapisanych dwóch słów kluczowych i odpowiada komunikatem:

```
Expected:  expression or end-of-statement
```

oznacza

Oczekiwane (w tym miejscu): wyrażenie lub koniec instrukcji.

Oznacza to, że za słowem kluczowym CLS powinien być koniec linii programu. (W systemie QBasic słowem *statement* określa się zarówno instrukcję, jak procedurę wbudowaną w system.)

Potwierdź zapoznanie się z komunikatem naciśnięciem klawisza *Enter* i usuń słowo PRINT.

Uwaga. W niektórych wersjach systemu QBasic komunikat ten zostanie pokazany dopiero po próbie uruchomienia programu.

Gdybyś dopisał PRINT przed CLS, to zobaczyłbyś komunikat

```
Syntax error
```

oznaczający

Błąd składniowy

wynikający z tego, że system nie dopuszcza takiej składni instrukcji programu. Zauważ, że wynika to z potraktowania CLS jako słowa kluczowego, ponieważ gdybyś zamienił CLS na CL lub CLSS, to takich komunikatów system by nie podawał (uznawszy zapewne CLSS za nazwę zmiennej i instrukcję drukowania tej zmiennej za prawidłową).

Nie zapisuj tak zmodyfikowanego pliku i zakończ edycję.

Uwaga. Wstępna kontrola linii programu w trakcie jego edycji nie wykrywa wszystkich błędów. Mogą one być wykryte dopiero w trakcie jego tłumaczenia i wykonywania.

9.3.3. Korzystanie z pomocniczego opisu systemu

Inną formą wspomagania edycji programu, a także posługiwania się systemem QBasic, jest sprzężenie edytora z programem pomocy. Jeżeli najedziesz kursorem na słowo będące instrukcją systemu, np. CLS lub PRINT, lub innym słowem kluczowym i naciśniesz klawisz *F1*, to program poda Ci informacje na temat każdego z tych terminów (po angielsku). Dla przykładu uczyń to w odniesieniu do słowa PRINT (występuje ono w każdym z omawianych dotychczas programów; możesz też zacząć edycję nowego pliku i je napisać).

Prawidłowy zapis instrukcji jest podawany z użyciem symboli, które mogą Ci się wydać niezrozumiałe. Nawiasy kwadratowe zawierają takie elementy, które mogą być użyte, lecz nie muszą. Jeżeli może być użyty jeden z kilku elementów pewnego typu, to takie elementy są objęte nawiasami klamrowymi (jako zbiór), a od siebie są oddzielone pionowymi kreskami. Zwróć uwagę na fragment zapisany następująco:

```
[{;|,}]
```

Oznacza on, że można używać (w danym miejscu) albo przecinka, albo średnika. Czy tak samo je zrozumiałeś?

Podany kilka wierszy niżej opis wyjaśnia znaczenie poszczególnych symboli. Strefa drukowania następnego elementu zaczyna się bezpośrednio po ostatnim wydrukowanym elemencie, jeżeli w instrukcji jest po nim średnik, zaś na początku następnej 14 znakowej strefy, jeżeli jest zakończony przecinkiem.

Nawet jeżeli nie znasz angielskiego, możesz wynieść korzyść z tej formy pomocy, ponieważ często w opisie poszczególnych haseł podane są przykłady prawidłowego użycia danych słów kluczowych. Kilka terminów angielskich znajdziesz też w słowniczku zamieszczonym na końcu książki.

Informacja może obejmować więcej niż jeden ekran. Do przeniesienia kursora do okna pomocy służą klawisze *Shift-F6*, a do przewijania tekstu — klawisze kursora oraz *PgDn* i *PgUp*. Do przesuwania kursora między wyróżnionymi hasłami służy klawisz *Tab*.

Do trybu edycji powracasz naciskając klawisz *Esc*.

Z programu pomocy możesz korzystać także bez wskazywania z góry na konkretny termin, uaktywniając pole Help w menu. Oprócz znanej Ci już pomocy na zadany temat (Topic) masz wtedy do wyboru: **indeks alfabetyczny haseł** (Index) oraz opis tematów (Contents). Ponadto możesz dowiedzieć się, jak używać programu pomocy (Using Help). Dodatkową możliwością jest uzyskanie informacji o wersji systemu QBasic (About...).

Informację z okna pomocy, na przykład program zawierający konkretną instrukcję, możesz kopiować do okna edycyjnego posługując się poleceniami edytora w sposób, do jakiego zapewne już się przyzwyczaiłeś: powinieneś zaznaczyć tekst (na przykład *Shift*-klawisze kursora), skopiować do schowka (*Ctrl-Ins*) i skopiować ze schowka do okna edycyjnego (*Shift-Ins*). Polecenia edytora i odpowiadające im kombinacje klawiszy znajdziesz pod hasłem Editing Keys.

Przypuśćmy, że znając nazwę instrukcji warunkowej (IF) pragniesz zobaczyć, w jaki sposób powinna być używana.

Ćwiczenie 9-4

Uruchom system QBasic rozpoczynając edycję nowego pliku, napisz słowo IF, naciśnij klawisz *F1*, „wejdź" do okna pomocy, zaznacz przykładowy program i skopiuj go do schowka. Wyjdź z programu pomocy, skasuj słowo IF i skopiuj zawartość schowka. Czy otrzymałeś taki sam program (lub fragment), jaki zaznaczyłeś w oknie pomocy?

9.3.4. Pomoc systemu w uruchamianiu programów

Może się zdarzyć, że podczas wykonywania Twojego programu wystąpi błąd. W takiej sytuacji pojawi się komunikat: `Run-Time Error` *n* (*błąd wykonania n*) albo okno ze słownym określeniem błędu. Rodzaj błędu określonego numerem *n* znajdziesz w pomocy (Contents, Run-Time Error Codes). Przytaczanie tu polskiego znaczenia wszystkich wersji komunikatów nie jest możliwe, niemniej znasz już tyle angielskich słów z dziedziny informatyki, że dasz sobie radę (ewentualnie z pomocą słownika).

Jeżeli nie zauważysz w Twoim programie błędu odpowiedzialnego za sygnalizowany błąd wykonania, to uruchamiaj program z użyciem tzw. **debugera** (rozdz. 8.5.5). Dostęp do jego poleceń uzyskujesz przez uaktywnienie okna menu Debug; niektóre z jego poleceń są też dostępne bezpośrednio z trybu edycji po naciśnięciu klawiszy funkcyjnych.

Pomoc w uruchamianiu Twoich programów (i znajdywaniu błędów) polega na następujących ingerencjach w bieg programu:

- Wykonywanie programu instrukcja po instrukcji — polecenie Step (*krok*; wywoływane również klawiszem *F8*). Jest też odmiana tego polecenia: Procedure Step (*F10*), traktująca podprogramy jako pojedyncze instrukcje. Działanie przejawia się tym, że po każdym naciśnięciu klawisza wykonywana jest tylko jedna instrukcja programu. Program się zatrzymuje i możesz obejrzeć ekran wynikowy (*F4*).
- Śledzenie biegu programu — polecenie Trace (ślad). Jego działanie przejawia się tym, że wykonywana aktualnie instrukcja jest podświetlana na ekranie edycyjnym. Działanie programu można przerwać naciśnięciem klawiszy *Ctrl-Break*.
- Wykonywanie programu do zaznaczonej instrukcji. Do zaznaczania instrukcji, a również do usuwania zaznaczenia, służy polecenie Toggle Breakpoint (wstaw/usuń punkt zatrzymania), jak również klawisz funkcyjny *F9*. Zaznaczona instrukcja jest przedstawiana na ekranie innym kolorem. Jeżeli uruchomisz taki program, to wykona się on do zaznaczonej instrukcji i zatrzyma. Otworzy się wtedy w dolnej części ekranu pomocnicze okno o nazwie Immediate (natychmiastowy), w którym możesz odczytać wartości zmiennych. W razie potrzeby

przechodzisz do tego okna, naciskając klawisz *F6* (polecenie Window, podpowiadane w dolnej części ekranu) i wracasz w ten sam sposób.

- Zmiana kolejności wykonywania instrukcji po zatrzymaniu programu. Służy do tego polecenie Set Next Statement (następna instrukcja). Posługujesz się nim w ten sposób, że najpierw przesuwasz kursor do miejsca, od którego chciałbyś wykonywanie programu kontynuować, a następnie wydajesz to polecenie.

Ćwiczenie 9-5

Odczytaj program QB-2.

A. Wykonaj go z włączoną opcją śledzenia.
B. Zaznacz wybraną instrukcję, na przykład drugą instrukcję PRINT, jako punkt zatrzymania i uruchom program. Po wykonaniu części instrukcji (będziesz proszony o wprowadzenie liczby) program się zatrzymuje i zaznaczona instrukcja zostaje podświetlona. Przejdź do pomocniczego okna Immediate, naciskając klawisz *F6*, i następnie napisz oraz wykonaj polecenie `PRINT imie$, wiek`. Czy wydrukowane wartości odpowiadają Twoim oczekiwaniom?
C. Powróć do okna edycyjnego (*F6*) i poleć kontynuowanie programu (polecenie Continue — klawisz *F5*). Czy wartości wydrukowane na końcu programu są zgodne z Twoimi oczekiwaniami?

9.4. Zmienne i stałe

9.4.1. Nazwy zmiennych

Każda zmienna ma nazwę. Nazwa w systemie QBasic może się składać z 40 znaków: liter, cyfr i kropek, przy czym pierwszy znak musi być literą. Zmienne nie mogą nosić nazw zastrzeżonych dla instrukcji, deklaracji, funkcji i innych słów kluczowych. Do poprawnych nazw należą zatem:

```
ala   J23   x1Y2   czytelnik.wiek
```

natomiast nazwy:

```
1ala  x1-Y2   cls  aaaabbbbccccdddddeeeeffffffgggghhhhi
```

nie są poprawne, pierwsza z nich zaczyna się bowiem od cyfry, druga zawiera niedozwolony w nazwie znak *minus*, trzecia jest nazwą instrukcji (CLS), a czwarta jest za długa.

Nazwa zmiennej może się odnosić zarówno do zmiennej prostej (pojedynczej), jak i do tablicy zmiennych. W przypadku zmiennych prostych sam fakt użycia w tekście programu jej nazwy jest równoznaczny z jej

deklaracją. Zauważ, że do tej pory zmienne występujące w programach nie były deklarowane. Tablice wymagają jawnej deklaracji (reguła ta ma wyjątek w odniesieniu do niewielkich tablic, którego jednak nie będziemy tu uwzględniać).

Ostatnim znakiem nazwy może być znak specjalny spoza wymienionych, np. $ lub % (patrz dalej), służący do określania typu zmiennej. Zmiennej *x$* już używałeś do przechowywania tekstu. Wszystkie używane do tej pory nazwy składały się z jednej lub kilku liter.

System traktuje wielkie litery w nazwach jako identyczne z małymi. Na przykład nazwy:

```
mat   Mat   MAT
```

system będzie odnosił do jednej zmiennej, a nie trzech różnych zmiennych.

Programy są łatwiejsze do sprawdzania, jeżeli zmienne noszą nazwy, które kojarzą się z reprezentowaną przez zmienną wielkością. W algorytmach stosowane były często właśnie takie nazwy, na przykład MNOZNIK, ODPOWIEDŹ, NUMER, a nawet nazwy zawierające w sobie kilka słów lub ich fragmentów, np. DOBREODP czy NUMERLOS. Możesz identycznych lub podobnych nazw użyć w programach realizujących te algorytmy. Pamiętaj o konieczności zastąpienia polskich liter innymi, np. ich łacińskimi odpowiednikami.

Ponieważ w systemie QBasic nazwy instrukcji są pisane wielkimi literami, więc nazwy zmiennych możesz pisać małymi, żeby program był przejrzystszy.

Do dobrych zwyczajów stosowanych w odniesieniu do nazw składających się z kilku słów znaczących nie oddzielonych od siebie żadnym znakiem jest stosowanie wielkich liter na początku każdego kolejnego słowa, np.

```
DobreOdp   NumerLos
```

Jeżeli w jakimś programie stosujesz taką konwencję, to możesz ją rozszerzyć na nazwy składające się z jednego słowa. Czuj się jednak wolny w wyborze konwencji stosowania wielkich i małych liter w nazwach zmiennych.

9.4.2. Typy zmiennych prostych

Zmienne służą do przedstawiania liczb (zmienne liczbowe) i tekstów (zmienne łańcuchowe). Zmienne służą przede wszystkim do przedstawiania pojedynczych wielkości, tj. jednej liczby lub jednego tekstu. Takie zmienne nazywane są **zmiennymi prostymi**. **Zmienne złożone** mogą przedstawiać wiele wielkości; przykładem są tablice (a także rekordy, takie jak w bazach danych).

Tabela 9.1. Typy zmiennych liczbowych

Typ zmiennej		Przyrostek	Przykłady	Wartość maksymalna
Integer	Całkowita	%	numer%	32767
Long-integer	Całkowita długa	&	mieszk&	2147483647
Single-precision	Rzeczywista	!	odl!, odl	$3.402 * 10^{38}$
Double-precision	Rzeczywista długa podwójnej precyzji	#	kappa#	$1.798 * 10^{308}$

Zmienne liczbowe

W programach można używać kilku typów zmiennych liczbowych (tabela 9.1); dwa z nich służą do zapisywania liczb całkowitych i dwa — rzeczywistych, z częścią ułamkową. Różnią się liczbą bajtów użytych do zapisu liczby, a więc także zakresem liczb dających się zapisać i precyzją zapisu liczb rzeczywistych. Typ zmiennej deklaruje się używając jednego z wymienionych przyrostków jako ostatniego znaku nazwy (patrz nazwy przykładowe).

Wszystkie używane do tej pory zmienne liczbowe były, jak widać, zmiennymi rzeczywistymi (pojedynczej precyzji).

Zmienne łańcuchowe (tekstowe)

Zmienne łańcuchowe zawierają ciągi (łańcuchy) znaków: liter, cyfr, znaków przestankowych, znaków spacji, znaków specjalnych ($, %, & i in.). Nazwa zmiennej łańcuchowej musi zawierać znak $ jako ostatni znak nazwy (jest to równoważne deklaracji typu). Zmienna *imie$* używana w programie QB-2 była więc zmienną łańcuchową. Zmienna łańcuchowa może zawierać do 32767 znaków.

Uwaga. Nazwy różniące się przyrostkiem, np. takie jak: *x%, x&, x!, x#, x$*, odnoszą się do różnych typów zmiennych i można ich używać jednocześnie bez obawy, że system je pomyli (może natomiast pomylić je autor programu). Jeżeli w nazwie zmiennej nie użyjesz żadnego przyrostka, to program zakłada, że zmienna ta służy do przedstawiania liczby rzeczywistej (*single-precision*). Z tego powodu w tab. 9.1 podano w tym wierszu dwie przykładowe nazwy: *odl!* oraz *odl*, traktowane przez program jako równoważne nazwy tej samej zmiennej.

Wybierając typ zmiennych kieruj się wiadomościami i wskazówkami z rozdz. 8.5.2.

9.4.3. Nadawanie wartości zmiennym

Zmiennym możesz w programie nadawać wartości. W arkuszu kalkulacyjnym czyniłeś to wpisując liczby do konkretnej komórki, tu natomiast posłużysz się instrukcją LET (niech — w odniesieniu do czynności). Na przykład

```
LET x = 2.5
```

jest instrukcją nadania zmiennej liczbowej x wartości 2.5, natomiast

```
LET napis$ = "Witam Ciebie ponownie!"
```

instrukcją nadania zmiennej łańcuchowej *napis$* wartości w postaci ciągu 22 znaków objętego cudzysłowami (od wielkiej litery W do wykrzyknika, łącznie ze spacjami).

Samo słowo LET może być pominięte, zatem taki sam skutek osiągniesz pisząc obie instrukcje bez niego.

```
x = 2.5
napis$ = "Witam Ciebie ponownie!"
```

Uwaga. Zapis $x = 2.5$ nie jest tu równaniem, którego rozwiązanie należałoby obliczyć, ani relacją, która jest spełniona lub nie zależnie od wartości zmiennej x, lecz **operacją przypisania wartości**: zmiennej x występującej po lewej stronie znaku równości zostaje nadana nowa wartość obliczona ze wzoru umieszczonego po prawej stronie znaku (patrz też rozdział 8.4).

Zmienne mają wartości początkowe nadane przez system w chwili uruchamiania programu.

Ćwiczenie 9-6

A. Napisz program QB-4 lub odczytaj go z pliku:

Program QB-4

```
REM Program QB-4 - nadawanie wartosci zmiennym
REM   tu jest miejsce na Twoj komentarz
CLS
PRINT x, napis$
LET x = 2
PRINT x, napis$
LET napis$ = "Nowa wartosc zmiennej tekstowej"
PRINT x, napis$
END
```

B. Wykonaj program.

Czy wynik ma postać następującą?

```
0
2
2          Nowa wartosc zmiennej tekstowej
```

C. Jeżeli program pisałeś, to zapisz go na dysku.

Na początku programu zmienna liczbowa x ma wartość zerową (nadawaną przez program każdej zmiennej liczbowej), a zmienna łańcuchowa *x$* jest pusta.

Na końcu programu QB-4 jest umieszczona instrukcja END, wskazująca jego koniec. Program QBasic rozpoznaje koniec programu również bez instrukcji END, nie jest więc ona teraz konieczna. Będzie przydatna, jeżeli działanie programu miałoby zostać zakończone przed dojściem do ostatniej instrukcji (a będzie konieczna, gdy z programu będą wyodrębnione podprogramy).

Uwaga. Nadając zmiennym wartości, powinieneś pamiętać o ich typie. Nie nadawaj zatem zmiennej całkowitej *i%* wartości rzeczywistej 2.5 albo zmiennej łańcuchowej *napis$* wartości liczbowej. Jeżeli spróbujesz wykonać program zawierający instrukcję:

```
LET napis$ = 3
```

to system zasygnalizuje niezgodność typów komunikatem:

```
TYPE MISMATCH
```

Jeżeli zaś wykonasz instrukcję:

```
LET i% = 2.5
```

to drukując następnie wartość zmiennej *i%* (instrukcją `PRINT i%`) przekonasz się, że stała 2.5 została zaokrąglona (obcięta) do wartości całkowitej 2, i to bez komunikatu ostrzegawczego. Tego rodzaju niedopatrzenia mogą prowadzić do błędnych wyników.

Jeżeli zostaje przekroczony zakres liczb, to program sygnalizuje **nadmiar** (*overflow*). Możesz sprawdzić, jak taki komunikat wygląda, wykonując program pomocniczy zawierający instrukcję:

```
LET i% = 40000
```

9.4.4. Zapisywanie liczb

System analizuje wstępnie każdą linię programu bezpośrednio po jej napisaniu (po przejściu do innej linii). Analiza obejmuje także zapisywane liczby

(zapisywałeś je po prawej stronie instrukcji przypisania). Stwarza to dobrą okazję do sprawdzenia, jak duże liczby system przyjmuje, a jakich nie.

Dotychczas zapisywałeś liczby całkowite lub liczby rzeczywiste z kropką dziesiętną. Bardzo duże lub bardzo małe liczby rzeczywiste możesz zapisywać w postaci wykładniczej. Zapisujesz wtedy mantysę zawierającą cyfry znaczące z kropką dziesiętną, np. 1.23456, oraz podajesz potęgę dziesięciu (całkowitą) posługując się symbolem E; np. 1.234E+12 lub 3.3467E–5. Pierwsza z tych liczb mogłaby być zapisana w postaci 1234000000000, druga jako 0.000033467.

Liczby tak duże, że wymagają typu podwójnej precyzji (tzn. większe niż $3,402*10^{38}$), należy zapisywać z użyciem symbolu D zamiast E, na przykład 1.23D+42.

Gdy zapisujesz stałą, wówczas system sam rozpoznaje jej typ. Na przykład stałą 12 zapisze jako liczbę całkowitą, 35000 — jako liczbę całkowitą długą (*long-integer*), 1.23 — jako liczbę rzeczywistą, a 1.23D+42 — jako liczbę rzeczywistą podwójnej precyzji.

Może on nawet ingerować w Twój zapis.

Liczbę całkowitą zbyt dużą, żeby mogła być reprezentowana jako liczba typu *long-integer*, np. 2222222222, system potraktuje jako liczbę rzeczywistą, i to podwójnej precyzji, co zaznaczy dodaniem przyrostka #. Wówczas użyje on postaci podwójnej precyzji, ponieważ wprowadzona liczba ma zbyt wiele cyfr znaczących, żeby można ją przedstawić dokładnie jako liczbę pojedynczej precyzji.

System zaingeruje również wtedy, gdy będziesz próbował zapisać liczbę zbyt dużą w zapisie wykładniczym, np. 1.23E+39. Będzie on oponował po zakończeniu pisania tej linii programu i będziesz musiał albo zmniejszyć liczbę do zakresu mieszczącego się w pojedynczej precyzji (np. 1.23E+38), albo użyć symbolu D zamiast E.

Zauważ też, że zapisując liczby rzeczywiste w postaci wykładniczej na oznaczenie symboli D lub E nie możesz użyć małych liter d, e.

9.4.5. Używanie stałych

Zapisywane w treści programu teksty (w cudzysłowach) i liczby system traktuje jako stałe. Używałeś już stałej tekstowej w Twoim pierwszym programie w instrukcji drukowania `PRINT "Witam Ciebie!"`. Zapisywałeś liczby po prawej stronie instrukcji przypisania.

Niekiedy jest wygodnie nadawać stałym nazwy. Gdybyś miał kilka razy wykonywać obliczenia z użyciem liczby π, to zamiast pisać za każdym razem 3.14159, mógłbyś zadeklarować stałą o nazwie *pi*, nadając jej jednocześnie podaną wartość. Wówczas we wzorach możesz używać nazwy *pi*.

Program QB-5

```
REM Program QB-5   Obliczanie obwodu kola
CONST PI = 3.14159
INPUT "Podaj promien kola ", r
PRINT "Obwod kola o promieniu "; r; "wynosi"; 2 * PI * r
```

Rozwiązanie takie ma zalety związane z użyciem pamięci komputera (co ma znaczenie przy większych programach), przede wszystkim jednak zmniejsza niebezpieczeństwo popełnienia błędu przez niejednakowe napisanie wartości stałej w różnych miejscach programu oraz czyni program bardziej przejrzystym, ułatwiając jego kontrolę i wyszukiwanie ewentualnych błędów.

Ćwiczenie 9-7

Zmodyfikuj program QB-5 tak, by wyznaczał pole koła.

9.5. Działania na zmiennych liczbowych

9.5.1. Operacje arytmetyczne

Na zmiennych i stałych liczbowych możesz wykonywać różne działania matematyczne, podobnie jak w arkuszu kalkulacyjnym. Oprócz czterech działań arytmetycznych: dodawania, odejmowania, mnożenia i dzielenia, możesz wykonywać potęgowanie (x^y) oraz dzielenie całkowite (tab. 9.2).

Tabela 9.2. Działania na zmiennych liczbowych

Działanie	Symbol	Przykład	Wynik
Dodawanie	+	3 + 5.5	8.5
Odejmowanie	−	17 − 12	5
Mnożenie	*	3 * 17	51
Dzielenie	/	9/4	2.25
Potęgowanie	^	3^2	9 (3^2)
Dzielenie całkowite	\	11\4	2
Reszta z dzielenia całkowitego	MOD	11 MOD 4	3

Dla przypomnienia: w dzieleniu całkowitym dzieli się liczbę całkowitą przez liczbę całkowitą. Wynik dzielenia jest zawsze liczbą całkowitą — oznacza ona, ile razy dzielnik mieści się w całości w dzielnej; ponadto z dzielenia może pozostać reszta. Na przykład przy dzieleniu 13 przez 5 wynik dzielenia całkowitego wynosi 2, a z dzielenia pozostaje reszta równa 3. W celu oswojenia się z zapisywaniem rzadziej używanych działań wykonaj ćwiczenie 9-8.