Programowanie komputerów

8

Wprowadzenie do programowania

8.1. Rozwiązywanie problemów z użyciem komputera

8.1.1. Informatyka a programy komputerowe

Znasz już różne programy komputerowe: zarówno te wchodzące w skład systemu operacyjnego, jak i te zaliczane do programów użytkowych. Wiesz już, jak je uruchamiać i niektórymi z nich umiesz się posługiwać. Zapewne umiałbyś wskazać możliwość stosowania ich do pewnych zadań praktycznych, z którymi się stykasz.

Poznając kolejny typ programu użytkowego zwróciłeś z pewnością uwagę na podobne elementy w działaniu różnych programów, i to tak różnych, jak edytory tekstów, arkusze kalkulacyjne i programy graficzne. Niezależnie bowiem od dziedziny, w jakiej się je stosuje, niemal w każdym występuje:

1) **wprowadzanie danych** z klawiatury, myszy, pliku dyskowego;
2) **przetwarzanie danych**;
3) **wyprowadzanie wyników** na ekran, do pliku, na drukarkę.

Listę urządzeń do wprowadzania danych i wyprowadzania wyników uzupełnisz z łatwością, choćby o skaner, głośnik i ploter.

Programy komunikacyjne nie przetwarzają danych (zmieniana jest jedynie postać fizyczna sygnału, np. w modemie lub karcie sieciowej), realizują natomiast:

4) **przesyłanie danych**.

Danymi są nie tylko słowa, liczby czy obrazy, lecz także polecenia wydawane programowi przez użytkownika, a również zdarzenia, jak np. naciśnięcie

klawisza. Dane są przedstawiane w sposób sformalizowany, co ułatwia ich przetwarzanie i przesyłanie. Z danymi jest związana **informacja**; mianem tym określa się znaczenie przypisywane danym.

Przetwarzanie danych polega na uporządkowanym wykonywaniu na nich operacji (jest określane także mianem przetwarzania informacji). Ogólnie biorąc, przetwarzanie danych może być wykonywane ręcznie przez człowieka lub automatycznie, za pomocą odpowiednich środków. W praktyce środkami tymi są komputery wyposażone w odpowiednie programy.

Informatyka jest dyscypliną naukowo-techniczną dotyczącą zagadnień związanych z przetwarzaniem danych za pomocą środków automatycznych. Podstawowymi zagadnieniami informatyki są:

- gromadzenie informacji,
- przetwarzanie informacji,
- udostępnianie informacji,
- przesyłanie informacji.

Z zagadnieniami tymi już się zetknąłeś, m.in. w bazach danych. Możesz spojrzeć na wprowadzanie danych jak na **gromadzenie informacji**, a na wyprowadzanie wyników — jak na **udostępnianie** jej; przykładem przetwarzania informacji jest jej sortowanie lub wybieranie tych rekordów, które odpowiadają wymaganiom. Przesyłanie informacji jest łatwo dostrzegalne w sieciach komputerowych, zwłaszcza rozległych, aczkolwiek można o nim mówić także w odniesieniu do transmisji między podzespołami komputera.

Informatyka może być Ci pomocna w rozwiązywaniu rozmaitych problemów, jeżeli występują w nich wymienione zagadnienia. Szczególnie dużo mógłbyś skorzystać z osiągnięć informatyki w zakresie przetwarzania danych, gdzie jej dorobkiem są liczne opracowane **algorytmy**, czyli reguły postępowania służące do rozwiązywania konkretnych zadań dla różnych zestawów danych, zapewniające otrzymanie rozwiązania w skończonej liczbie kroków.

8.1.2. Schemat rozwiązywania problemów w informatyce

Wyliczenie podane w rozdz. 8.1.1 pokazuje zarazem, w jaki sposób należy przedstawić problem, który chce się rozwiązać (lub zadanie, które chce się wykonać), żeby móc skorzystać z podejścia informatycznego:

1. W problemie należy wyodrębnić takie składniki, jak:
   - wprowadzanie informacji,
   - przetwarzanie informacji,
   - wyprowadzanie wyników,

   i każdy z nich przedstawić w sposób właściwy dla informatyki:
   - upewnić się, że informacja wprowadzana będzie miała formę da-

nych o konkretnej postaci akceptowanej przez komputer (np. liczb, tekstu, naciśnięć klawiszy, sygnałów i ruchów myszy), a w razie potrzeby dobrać odpowiednie urządzenie wejściowe (np. laboratoryjne urządzenie pomiarowe przystosowane do współpracy z komputerem);
- określić dokładnie sposób przetwarzania informacji (problem ten często jest uważany za najważniejsze zagadnienie informatyki), w szczególności upewnić się, że dostępne dane wystarczą do uzyskania prawidłowych wyników;
- określić sposób przedstawiania wyników za pomocą dostępnych urządzeń wyjściowych, a jeżeli nie jest to możliwe, to dobrać odpowiednie urządzenia wyjściowe.

2. Dobrać lub utworzyć program komputerowy, który (samoczynnie) **wykona zadania** wprowadzania informacji, przetwarzania jej i wyprowadzania wyników określone szczegółowo w punkcie 1.

Ogólnie biorąc można potrzebny program wykonać samodzielnie lub zlecić jego wykonanie albo posłużyć się programem gotowym. Opracowano wiele programów do konkretnego typu zadań (posługując się nimi nie trzeba nawet znać szczegółowo sposobu przetwarzania informacji, a jedynie cel jej przetwarzania). Są też programy bardziej uniwersalne, jak poznane przez Ciebie arkusze kalkulacyjne i systemy zarządzania bazami danych. Zauważ, że posługując się wymienionymi programami użytkowymi, a w szczególności wykonując niektóre ćwiczenia — w istocie rozwiązywałeś już pewne problemy.

Zapewne wykonywałeś ćwiczenia 5-18, 5-19 i 5-21, lub przynajmniej niektóre z nich. Tworzyłeś wówczas modele arkusza kalkulacyjnego dla problemów rozwiązywania równania liniowego $ax + b = cx + d$ (RLIN1 i RLIN2) oraz równania kwadratowego $ax^2 + bx + c = 0$ (RKWADR1 i RKWADR2). Modele te umożliwiały:
- wprowadzanie danych — parametry a, b, c i ewentualnie także d były umieszczone w konkretnych komórkach arkusza, a wprowadzanie następowało w wyniku wpisania ich bezpośrednio do tych komórek (w sposób przyjęty w programie arkusza);
- przetwarzanie danych, odpowiadające rozwiązaniu równań zgodnie ze znanymi ogólnie algorytmami — obliczenia były wykonywane samoczynnie; sposób wykonywania obliczeń określiłeś za pomocą wzorów, a w ćwiczeniu 5-21 dodatkowo także za pomocą instrukcji wykonywanych zależnie od spełnienia badanych warunków;
- przedstawianie wyników — były to liczby i komunikaty pojawiające się w odpowiednich komórkach arkusza.

Modele RLIN1 i RKWADR1 działały poprawnie dla takich wartości parametrów, przy których rozwiązania równań istniały i były jednoznaczne, natomiast RLIN2 i RKWADR2, zawierające badanie odpowiednich warunków, umożliwiały otrzymanie wyników przy dowolnych danych, z tym że przy braku rozwiązań bądź ich niejednoznaczności wynik miał postać komunikatu tekstowego.

Ćwiczenie 8-1

Przeanalizuj wykonane przez Ciebie ćwiczenie (wskazane przez nauczyciela lub wybrane przez Ciebie) dotyczące:
a) pracy z edytorem tekstów,
b) wykonywania obliczeń z użyciem arkusza kalkulacyjnego,
a) tworzenia i obsługi bazy danych,
wyodrębniając i opisując wymienione składniki (wprowadzania, przetwarzania i wyprowadzania informacji).

Uwaga. Jeżeli opis sposobu przetwarzania informacji byłby obszerny, to ogranicz się do najważniejszych jego elementów.

Nawet na same programy użytkowe czy narzędziowe, jakich używasz, możesz spojrzeć jako na rezultat rozwiązywania problemu postawionego przed ich projektantami i wykonawcami.

Ćwiczenie 8-2

A. Przeanalizuj program, który lubisz najbardziej, lub program wskazany przez nauczyciela i wydziel w nim trzy składniki funkcjonalne: wprowadzanie informacji, przetwarzanie jej oraz wyprowadzanie wyników.
B. Określ, jakie zadania cząstkowe program wykonuje (jeżeli jest ich dużo, to wypisz kilka najważniejszych) i spróbuj odtworzyć, jak mógł być sformułowany problem postawiony przed twórcami programu.

W tej części książki zajmiemy się tworzeniem programów służących do wykonywania zadań określonych w p. 1 przedstawionego schematu rozwiązywania problemów.

8.1.3. Kiedy jest potrzebne samodzielne tworzenie programów

Poznane przez Ciebie programy użytkowe stwarzają spore możliwości wykonywania rozmaitych zadań, ale tylko w obrębie swojego obszaru działania, a więc przede wszystkim arkuszy kalkulacyjnych i baz danych. Możesz się

zetknąć z problemami wymagającymi użycia takich działań, jakich typowe programy użytkowe nie zapewniają.

Na przykład możesz chcieć odczytać z pliku ASCII napisany przez koleżankę tekst, w którym polskie litery są kodowane w taki sposób, jakiego stosowany przez Ciebie edytor tekstów nie zna.

W szkole bywa potrzebne przygotowanie programów wspomagających nauczanie jakiegoś przedmiotu, rozwiązywanie zadań lub sprawdzanie wiadomości — do samodzielnego użycia przez uczniów.

Może też zaistnieć potrzeba przedstawienia w nietypowej postaci graficznej wyników eksperymentu fizycznego przeprowadzonego w pracowni szkolnej lub symulowania zjawiska fizycznego z przedstawianiem jego przebiegu na ekranie. Może być potrzebne przekształcenie pliku z danymi (na przykład zapisanymi przez automatyczne urządzenie pomiarowe) do postaci nadającej się do odczytania go przez program arkusza kalkulacyjnego.

Przechowywanie dużej ilości plików na dyskietkach wymaga porządnego gromadzenia i aktualizowania informacji o nich. Rozsądną decyzją jest korzystanie z gotowego systemu zarządzania bazami danych, w którym łatwo będzie wykonywać operacje na zgromadzonych informacjach i przedstawiać wyniki. Najtrudniejszym problemem staje się wtedy samo wprowadzanie informacji. Wprowadzanie ręczne jest pracochłonne, na jednej dyskietce może być przecież kilkadziesiąt, a nawet kilkaset plików; ponadto stanowi sposobność do popełnienia błędów. Dobrym wyjściem jest więc opracowanie programu, który odczytywałby z dyskietki nazwy plików i ich parametry (wielkość, datę i tak dalej) oraz tworzył plik, w którym potrzebne informacje byłyby zapisane w taki sposób, żeby można je było odczytać w systemie zarządzania bazami danych. Bezpośrednie odczytywanie z dyskietki informacji o zapisanych plikach jest trudne, można jednak posłużyć się plikiem tworzonym za pomocą polecenia `DIR` (rozdział 2.4.7). Problem sprowadzałby się wówczas do wybrania z takiego pliku informacji potrzebnych w bazie danych oraz zapisania ich w pliku wyjściowym w odpowiedni sposób.

Do wielu zadań, zwłaszcza często spotykanych, zostały opracowane programy; jednym z wyjść jest więc poszukanie odpowiedniego gotowego programu. Nie zawsze kupno takiego programu jest uzasadnione; na przykład gdy tekst koleżanki jest krótki, to prościej jest przepisać go od nowa lub poprawić litery ręcznie (posługując się na przykład edytorem Edit). Ponadto wyszukanie odpowiedniego programu gotowego może zabrać znacznie więcej czasu niż napisanie samemu programu, który zamieniałby kody polskich liter ze standardu stosowanego w tekście na standard akceptowany przez Twój edytor.

Umiejętność tworzenia programów może Ci się zatem przydać np. do napisania (prostego) programu dla własnych doraźnych potrzeb. Co ważniejsze, nawet sama wiedza o tworzeniu programów może okazać się przydatna w przyszłości, gdy np. widząc potrzebę opracowania programu do konkretnego zadania zlecałbyś jego napisanie.

Tworzenie programów jest jedną z głównych dziedzin informatyki, nazywaną **programowaniem**.

8.2. Podstawowe wiadomości o programowaniu

O tworzeniu programów wiesz niewiele. Wprawdzie już tworzyłeś proste programy w postaci plików typu BAT (rozdz. 2.5.2), ale zapisywałeś w nich polecenia systemu operacyjnego, a więc odwoływałeś się do istniejących już programów. Tu jednak chodzi o tworzenie **programów wykonywalnych**, czyli programów zawierających rozkazy bezpośrednio wykonywane przez (mikro)procesor — tzw. **rozkazy komputerowe**. Po stosowanych w systemie DOS rozszerzeniach EXE i COM umiesz rozpoznawać pliki dyskowe, w których są zawarte programy. Nie umiesz jednak ich tworzyć.

Celem tego rozdziału jest wprowadzenie Ciebie w dziedzinę programowania, czyli tworzenia programów. Będziesz mógł ją wtedy lepiej ocenić. Jeżeli Cię zainteresuje albo przekonasz się o jej przydatności praktycznej dla Ciebie, będziesz mógł kontynuować poznawanie jej na kolejnych przedmiotach informatycznych (jeżeli są w programie nauczania) lub na innych formach zajęć zorganizowanych przez szkołę. Mając dostęp do komputera i programów, możesz też uczyć się samodzielnie. Jeżeli nie zainteresuje Cię obecnie, to i tak będziesz mógł do niej powrócić w przyszłości, o ile uznasz to za potrzebne. Inna korzyść, jaką wyniesiesz, wiąże się z umiejętnością analizowania problemów i formułowania sposobów ich rozwiązywania za pomocą programów.

8.2.1. Języki programowania

Programowanie komputerów obejmuje:

a) projektowanie programów,
b) zapisywanie programów,
c) testowanie programów.

Programy wykonywalne, którymi się posługujesz, były przez kogoś lub przez zespół osób zaprojektowane, zapisane i zapewne dość długo testowane.

Gdy spojrzysz, jak duże są pliki zawierające te programy, i zastanowisz się, że zawierają nieznane Ci rozkazy komputerowe (których zapewne trzeba

byłoby się nauczyć), to może Ci się wydawać, że nakład pracy potrzebny na poznanie programowania jest ogromny. Byłoby tak rzeczywiście, gdyby trzeba było posługiwać się bezpośrednio rozkazami komputerowymi, i to w ich naturalnej postaci **kodu maszynowego**, tj. bajtów zapisywanych jako kombinacje zer i jedynek. Tworzenie programów komputerowych jest jednak znacznie ułatwione dzięki opracowaniu:

- **języków programowania**, służących do konstruowania programów komputerowych,
- **systemów programowania**, czyli oprogramowania służącego do opracowywania i używania programów zapisanych zgodnie z regułami danego języka programowania.

Języki programowania określają reguły formułowania (zapisywania) programu. Przy tworzeniu programów nie trzeba znać rozkazów komputerowych (rozkazów dla mikroprocesora), ponieważ wystarczy posługiwać się rozkazami języka programowania.

Przetłumaczenie programu sformułowanego zgodnie z regułami języka programowania na rozkazy komputerowe wykona program wchodzący w skład systemu programowania. Również ewentualne błędy w programie (przynajmniej niektóre) mogą zostać przez system programowania wykryte i wskazane.

Opracowano języki programowania o różnym stopniu szczegółowości i podobieństwa do języka naturalnego i do sposobu rozumowania człowieka. Mówi się w związku z tym o **poziomach języka programowania**. Na najniższym poziomie jest język komputerowy, składający się z rozkazów komputerowych w postaci „zerojedynkowej".

Zetkniesz się zapewne z pojęciem asemblera. Jest to **język programowania niskiego poziomu**, którego każdy rozkaz odpowiada jednemu rozkazowi komputerowemu; od języka komputerowego różni się tym, że jego rozkazy są zapisywane w sposób symboliczny, łatwiejszy do zapamiętania przez człowieka i kojarzenia. Językami niskiego poziomu nie będziemy się tu zajmować.

Języki wysokiego poziomu używają sformułowań podobnych do stosowanych w języku naturalnym (zazwyczaj angielskim); wzory matematyczne zapisujesz w sposób podobny do stosowanego w matematyce (albo w arkuszu kalkulacyjnym). Wymagają one znacznie mniejszego nakładu pracy przy tworzeniu programu, ponieważ jednemu rozkazowi języka wysokiego poziomu może odpowiadać kilka lub kilkadziesiąt rozkazów języka komputerowego. Są one uniwersalne w tym znaczeniu, że ich konstrukcje są niezależne od komputera i stosowanego systemu operacyjnego. Przykładem języka wysokiego poziomu jest Pascal.

Wśród opracowanych języków programowania są **języki problemowe**, odpowiadające pewnym klasom zastosowań, np. język zapytań SQL stosowany w dziedzinie relacyjnych baz danych. Są też języki uniwersalne, nie ograniczone do żadnej dziedziny, np. Pascal.

Do uniwersalnych języków programowania należy także Basic. W odróżnieniu od Pascala nie jest on zdefiniowany w formalny sposób, nie ma też jednego ogólnie przyjętego standardu języka Basic, niemniej jest on popularny. Programy umożliwiające tłumaczenie i wykonywanie programów źródłowych napisanych w języku Basic są stosowane w wielu mikrokomputerach osobistych, w tym i IBM PC. Basic stanowi podstawę dla nowoczesnych języków programowania, jak Visual Basic.

Język C służy nie tylko do opracowywania programów, lecz także do tworzenia systemów operacyjnych. Znany jest z tego, że przy jego użyciu zostały napisane podstawowe części systemu operacyjnego UNIX. Jest on jednym z najbardziej uniwersalnych języków programowania. W zastosowaniach profesjonalnych zajmuje pozycję dominującą.

Są także języki programowania związane z konkretnymi programami użytkowymi, np. język używany do tworzenia programów w systemie zarządzania bazami danych dBase.

Zastanawiasz się zapewne, czy warto lub czy należy uczyć się kilku języków programowania? Na obecnym etapie nie jest to celowe. Celem jest raczej poznanie typowych instrukcji języków programowania (stosowanych w różnych językach) i samego podejścia do tworzenia programu.

W przyszłości możesz być postawiony w sytuacji, w której powinieneś utworzyć (prosty) program, posługując się konkretnym językiem programowania — być może innym niż ten, którego uczyłeś się w szkole. Rozpoznanie w nim podobnych instrukcji ułatwi Ci to zadanie. Możesz też ocenić przydatność różnych języków do opracowania potrzebnego programu.

8.2.2. Systemy programowania

System programowania umożliwia edycję programu sformułowanego w konkretnym języku programowania oraz przetłumaczenie go i uruchomienie odpowiadającego mu programu wykonywalnego. System wskazuje też w razie potrzeby błędy programu uniemożliwiające poprawne przetłumaczenie go na program komputerowy. Może ponadto wykonywać różne operacje pomocnicze.

Warto mieć ogólne wyobrażenie o różnych typach systemów programowania, nawet jeżeli w pracowni będziesz posługiwać się tylko jednym. Podstawowe rozróżnienie wiąże się ze sposobem tłumaczenia programów. Program tłumaczący programy zapisane w języku programowania na programy

komputerowe (lub na programy zapisane w innym języku programowania) jest zwany ogólnie **translatorem** (tłumacz). Program sformułowany w języku programowania stanowi dane dla translatora i z tego względu jest nazywany **programem źródłowym**; translator przekształca go na **program wynikowy**. Program wynikowy może (ale nie musi) być programem wykonywalnym. Programy źródłowe określa się niekiedy mianem **kodu źródłowego**, zaś programy wynikowe, gdy są złożone z rozkazów komputerowych — mianem **kodu maszynowego** (określenie maszynowy pochodzi od maszyny matematycznej).

Są dwa podstawowe rodzaje programów tłumaczących:

- **Interpretery** (*interpreter* — tłumacz), tłumaczące poszczególne polecenia programu źródłowego na instrukcje mikroprocesora i przekazujące je od razu do wykonania przez mikroprocesor. Po zakończeniu wykonywania instrukcji podejmują one tłumaczenie następnego polecenia programu źródłowego.
- **Kompilatory** (*compiler* — zbierający fragmenty w całość), tłumaczące cały program źródłowy na program wynikowy, złożony z instrukcji mikroprocesora. Wykonywanie lub dalsze przekształcanie programu wynikowego może się rozpocząć dopiero po zakończeniu tłumaczenia przez kompilator całego programu źródłowego.

Z interpretowaniem poleceń już się zetknąłeś, obserwując jak system operacyjny wykonuje polecenia zapisane w pliku wsadowym o rozszerzeniu BAT jedno po drugim, natomiast o kompilatorach nie mówiliśmy jeszcze.

Interpretery tworzą plik dyskowy jedynie z programem źródłowym, natomiast z programem wynikowym, nie. Kompilatory zapisują na dysku oba rodzaje programów: zarówno źródłowy, jak wynikowy. Kompilowanie może odbywać się w jednym przebiegu programu tłumaczącego lub w kilku. Kompilator może zapisywać ponadto różne pliki pomocnicze (np. związane z pierwszym etapem kompilacji lub informujące o jej przebiegu). Do programu przetłumaczonego mogą być dołączane fragmenty programu z tzw. bibliotek (należących do systemu programowania).

Systemy programowania, którymi będziemy się posługiwać przy tworzeniu naszych programów, to Turbo Pascal i QBasic. Oba systemy są przeznaczone dla komputerów IBM PC.

Turbo Pascal jest popularnym systemem programowania umożliwiającym tworzenie programów źródłowych w języku Pascal, ich tłumaczenie na programy wykonywalne oraz wykonywanie. System ten kompiluje programy źródłowe (programy wykonywalne zapisuje w plikach typu EXE). Ma on postać rozbudowanego pakietu umożliwiającego szczegółową kontrolę działania uruchamianych programów.

System QBasic umożliwia tworzenie programów źródłowych w języku Basic, ich tłumaczenie i wykonywanie instrukcja po instrukcji (jest interpreterem). W porównaniu z Turbo Pascalem zajmuje niewiele miejsca. QBasic wchodzi w skład systemu operacyjnego MS DOS (można się nim posługiwać bez konieczności kupowania dodatkowych programów). Istnieją też kompilatory programów działających w systemie QBasic, ale nie są włączone do systemu operacyjnego MS DOS.

Programowanie w systemach QBasic i Turbo Pascal omówimy w rozdziale 9 i 10. W rozdziale 11 będziemy tworzyć programy i makropolecenia dla wybranych programów użytkowych oraz programy wsadowe, złożone z poleceń systemu operacyjnego DOS. Zetkniemy się też z językiem programowania nowego typu — Visual Basic.

Materiał zawarty w następnych rozdziałach powinien być potraktowany wybiórczo, zgodnie z zaleceniami nauczyciela. Wystarczy, jeżeli zapoznasz się z jednym z dwóch języków programowania: Basic lub Pascal, i odpowiednio z systemem programowania QBasic (rozdz. 9) lub Turbo Pascal (rozdz. 10). Zakres materiału też powinien być uzgodniony z nauczycielem, który może zalecić ograniczenie go albo zaproponować rozszerzenie o interesujące możliwości danego języka, pominięte w podręczniku. Jeżeli znasz akurat ten język programowania, który został wybrany przez nauczyciela dla Twojej grupy, a chciałbyś nauczyć się drugiego z nich, to zwróć się do nauczyciela. Rozdziały 9 i 10 mają podobny układ i zawierają te same lub podobne przykłady i ćwiczenia, mógłbyś więc towarzyszyć Twojej grupie pomimo tego, że posługiwałbyś się innym językiem programowania.

Nauczyciel może też zalecić posłużenie się innym językiem programowania niż wymienione dwa ze względu na jego wartości dydaktyczne; do zapisywania programów musisz wtedy posłużyć się odpowiednim opisem tego języka (rozdziały 9 i 10 możesz ominąć albo wykorzystać zawarte w nich ćwiczenia, realizując podobne programy w innym języku).

Korzystanie z rozdziału 11 jest zależne od wyposażenia pracowni szkolnej w programy. Nie muszą to być programy identyczne z opisanymi w tym rozdziale, ponieważ możliwe jest rozwiązywanie podobnych problemów także z zastosowaniem innych programów użytkowych.

Jeżeli wybór między systemami QBasic a Turbo Pascal będzie pozostawiony w pewnej mierze Tobie, to weź pod uwagę następujące czynniki: algorytmy przedstawione w książce mogą być zapisane i wykonane za pomocą każdego z nich; QBasic bardziej pomaga początkującemu użytkownikowi w unikaniu błędów i stawia mniejsze wymagania formalne; Turbo Pascal stwarza możliwości programowania na poziomie zaawansowanym (programowanie obiektowe, dostęp do zmiennych systemu operacyjnego) i lepiej

przygotowuje do kontynuowania nauki w tym kierunku; QBasic wprowadza do innych języków programowania, takich jak wspomniany Visual Basic oraz np. Access Basic, stosowany w systemie zarządzania bazami danych Access. W razie wątpliwości poradź się nauczyciela.

W dalszej części niniejszego rozdziału zajmiemy się pierwszym etapem tworzenia programu komputerowego, dotyczącym opracowania i przedstawienia sposobu rozwiązania zadania za pomocą programu — niezależnie od konkretnego języka programowania.

8.2.3. Etapy tworzenia programu komputerowego

Rozwiązując problem (rozdz. 8.1.2) określasz dane, sposób ich przetwarzania oraz wyniki, i na tej podstawie formułujesz **zadanie dla programu**. Niekiedy możesz zdecydować się na opracowanie kilku odrębnych programów współpracujących ze sobą (np. zapisane w pliku dyskowym wyniki działania jednego z nich mogą stanowić dane wejściowe dla innego); w takim wypadku określasz zadanie dla każdego z nich.

Przystępując do pracy nad tworzeniem programu, wiesz już zatem:

1) co właściwie program ma wykonywać,
2) jakich danych potrzebuje i w jakiej postaci ma je pobierać,
3) jak ma przetwarzać dane,
4) jakie wyniki ma dostarczać i w jakiej postaci,
5) czy ma się komunikować z użytkownikiem i jaką rolę mu przyznaje w trakcie działania.

W zależności od tego, jak szczegółowo problem został rozwiązany podczas analizowania go, zadanie jest już sformułowane mniej lub bardziej precyzyjnie. Na przykład może już być wiadomo, skąd mają być wprowadzane dane (z klawiatury czy z pliku dyskowego), gdzie mają być przedstawiane wyniki (na ekranie, w pliku dyskowym czy i tu, i tu) i w jakiej postaci (tekstowej czy graficznej; tylko wyniki ostateczne, czy także wyniki pośrednie).

Przykładowo, program do znajdowania pierwiastków trójmianu kwadratowego: $ax^2 + bx + c$ potrzebowałby jako danych trzech współczynników a, b i c tego trójmianu, jako wynik podawałby, czy trójmian ma pierwiastki, a jeżeli tak, to ile (1 czy 2) i o jakiej wartości. Dane miałby pobierać z klawiatury, po wypisaniu odpowiedniego komunikatu do użytkownika, wyniki zaś podawałby wyłącznie na ekranie. Rola użytkownika ograniczałaby się tu do uruchamiania programu i wprowadzania danych.

Jeżeli w sformułowaniu zadania brakuje tego rodzaju szczegółów, to decyzje podejmuje twórca programu. Jest wiele szczegółowych decyzji, które zazwyczaj nie są rozstrzygane na etapie ogólnego rozwiązywania problemu, lecz dopiero na etapie tworzenia programu.

W podanych dalej przykładach nie będziemy rozróżniać ściśle etapu rozwiązywania problemu i etapu tworzenia programu. W szczególności określanie sposobu przetwarzania danych będzie dla nas jednym z etapów tworzenia programu.

Kiedy już wiesz, co program ma robić, powinieneś rozstrzygnąć, jak ma to zrobić. Powinieneś sporządzić **algorytm**, czyli ściśle określić reguły rozwiązania zadania, i to w skończonej liczbie kroków. Jeżeli rozwiązanie opiera się na obliczeniach, to musisz podać, w jakiej kolejności i według jakich wzorów mają one być wykonywane.

Jeżeli pewne fragmenty obliczeń mają być powtarzane wielokrotnie, to powinieneś określić, w jaki sposób mają się kończyć, i upewnić, że się zakończą.

Następnym etapem jest **napisanie programu źródłowego** za pomocą konkretnego języka programowania (etap ten jest niekiedy określany mianem **kodowania programu**). Program ma realizować opracowany algorytm, zadania programu są jednak większe niż algorytmu, tak więc przy tworzeniu go muszą być podejmowane dodatkowe decyzje (w szczególności związane z odczytywaniem i przechowywaniem danych oraz wyprowadzaniem wyników). Wymaga to od twórcy programu zapoznania się z definicjami pojęć i konstrukcji używanych w danym języku oraz z zasadami formułowania poleceń i całych programów. Z reguły do napisania prostego programu wystarczy znajomość niewielkiej części wszystkich konstrukcji i rozkazów danego języka.

Sam program źródłowy jest zapisywany w pliku. Zazwyczaj jest to plik ASCII, który można przygotować pod dowolnym edytorem. Tekst programu podlega wtedy edycji na takich samych zasadach jak dowolny inny tekst. Często korzystniej jest przygotowywać tekst programu źródłowego pod edytorem należącym do stosowanego systemu programowania. Tekst programu może być wtedy poddawany wstępnej analizie, ułatwiającej dostrzeżenie ewentualnych pomyłek, a niekiedy nawet korygowany.

Trzeba jeszcze przygotować dane w postaci wymaganej przez program, jeżeli program ma odczytywać je z pliku.

Następnym etapem jest przetłumaczenie i wykonanie programu. W praktyce należy się liczyć z tym, że w programie mogą być błędy.

Zapamiętaj!
Zwykle programy zawierają błędy.
Twój program też może je mieć.

Wprawdzie masz szansę otrzymać prawidłowe wyniki już za pierwszym uruchomieniem programu, jednak doświadczenie wykazuje, że zazwyczaj jest inaczej, zwłaszcza w dużych, złożonych programach. Z tego powodu odpowiedzialni twórcy programów kładą bardzo duży nacisk na ich testowanie (przygotowują w tym celu wiele różnorodnych zestawów danych), a twórcy systemów programowania — na zapewnienie programistom pomocy w kontrolowaniu przebiegu programu i w wyszukiwaniu ewentualnych błędów.

Zespół czynności od napisania pierwszej wersji programu źródłowego do upewnienia się, że program działa poprawnie, określa się mianem **uruchomienia programu**. Jest to inne znaczenie niż odnoszące się do czynności systemu operacyjnego polegającej na umieszczeniu programu w pamięci komputera i zainicjowaniu wykonywania go; w konkretnym przypadku łatwo jest odróżnić, o które znaczenie chodzi.

Na etapie kompilacji i interpretacji programu źródłowego mogą być wykryte **błędy formalne**, powodujące że program tłumaczący nie jest w stanie przetłumaczyć tego programu na program wykonywalny. Program tłumaczący wypisuje wtedy (po angielsku) komunikat wskazujący na rodzaj błędu. Znalezienie przyczyn tego błędu i ich usunięcie może nie być łatwe, zwłaszcza dla początkującego programisty; może wymagać np. zwrócenia szczególnej uwagi na dokładne spełnienie formalnych wymagań języka, w rodzaju braku w jakimś miejscu średnika lub kropki (zwłaszcza w Turbo Pascalu).

Jeżeli program nie ma błędów formalnych, to zaczyna być wykonywany. Powinieneś otrzymać wyniki. Może się jednak zdarzyć, że program nie działa prawidłowo, sygnalizując **błędy wykonania**, na przykład brak pliku, z którego miał odczytać dane. Tego typu błędy są oznaczane niekiedy tylko numerem i trzeba zajrzeć do opisu programu, żeby dowiedzieć się, o jaki błąd chodzi.

Trudniejsza sytuacja występuje, gdy program nie sygnalizuje błędów, a jednak nie daje wyników lub daje niepoprawne. Jeszcze trudniejsza, gdy czasem daje wyniki poprawne, a czasem nie. Jeżeli zauważysz tego rodzaju błędy, to powinieneś przeanalizować działanie całego programu, sprawdzić początkowe wartości zmiennych, ich wartość po wprowadzeniu danych (może w pliku jest za mało danych i program brakujące dane odczytuje jako zera), nazwy plików z danymi i nazwy katalogów, a nawet sprawdzić poprawność samego algorytmu działania. Niektóre pakiety umożliwiają „podglądanie" programu w trakcie działania, co ułatwia wykrywanie ewentualnych nieprawidłowości.

Podsumowując podane informacje, wyliczymy **podstawowe etapy pracy nad programem:**

1. **Określenie zadania, do jakiego program ma służyć.**
2. **Opracowanie algorytmu realizowania zadania.**
3. **Utworzenie programu źródłowego w wybranym języku programowania.**
4. **Przygotowanie danych, uruchomienie i testowanie programu.**

To wszystko może wydawać Ci się trudne i skomplikowane. Przypomnij sobie jednak, że już napisałeś i uruchomiłeś prosty program (programy) w rozdz. 2.5.2, i to bez uczenia się pojęć i konstrukcji języków programowania, może więc jednak spróbujesz swoich sił...

8.3. Przykładowe problemy i zadania

Sformułujemy przykładowe zadania dla programów komputerowych. Jedne z nich mają charakter zadań szkolnych, inne wynikają z problemów praktycznych. Posłużą nam do zilustrowania poszczególnych etapów tworzenia programów.

Przedstawimy podstawowe elementy algorytmów rozwiązując zadania 8-1 ÷ 8-4.

Zadanie 8-1. Rozwiąż równanie kwadratowe $ax^2 + bx + c = 0$ dla danych wartości współczynników a, b, c.

Zadanie 8-2. Podaj wynik mnożenia podanej przez użytkownika liczby n przez wszystkie liczby całkowite od 2 do 10.

Zadanie 8-3. Odczytuj wprowadzane przez użytkownika liczby. Po odczytaniu kolejnej liczby obliczaj i podawaj wartość sumy wprowadzonych liczb oraz największej z wprowadzonych liczb. Kończ obliczenia po wprowadzeniu przez użytkownika liczby ujemnej.

Zadanie 8-4. Odczytaj n liczb. Wypisz je w kolejności od najmniejszej do największej.

Twój nauczyciel ze szkoły podstawowej poprosił Ciebie o pomoc w przygotowaniu programów ilustrujących wykonywanie obliczeń i pomocnego w sprawdzaniu prac. Jako przykładowe zadania wymienił:

Zadanie 8-5. Rozłóż na czynniki liczbę naturalną n podaną przez użytkownika.

Zadanie 8-6. Wyznacz największy wspólny dzielnik i najmniejszą wspólną wielokrotność dwóch liczb naturalnych.

Zadanie 8-7. Dodaj do siebie dwa ułamki (ilorazy liczb naturalnych); wynik przedstaw w postaci ułamka nieskracalnego.

Zadanie 8-8. Zilustruj wykonywanie działania dodawania pisemnego dla dwóch liczb całkowitych, z zaznaczeniem przeniesień występujących w dodawaniu poszczególnych pozycji (jedności, dziesiątek, setek itd.).

Analiza ruchu ciał sprawia trudności niektórym Twoim koleżankom i kolegom. Wiedząc że system programowania umożliwia zmianę koloru dowolnego punktu ekranu, pragniesz przygotować program ilustrujący ruchy ciał stanowiące przedmiot niektórych typowych zadań z matematyki i fizyki (zadanie 8-9), np. ruchu dwóch pojazdów jadących z różnymi prędkościami lub rzutu ukośnego.

Zadanie 8-9. Przedstaw na ekranie ruch punktu (lub kilku punktów), opisany danymi zależnościami: $x(t), y(t)$, gdzie x i y są współrzędnymi położenia punktu, a t oznacza czas.

Pragniesz wyobrazić sobie kształt funkcji dwóch zmiennych, nie dysponujesz jednak odpowiednim programem, umożliwiającym na przykład przedstawienie wartości funkcji w postaci powierzchni w przestrzeni trójwymiarowej. Będąc przyzwyczajonym do korzystania z map gotów jesteś przedstawić funkcję w postaci podobnej do stosowanej na mapach fizycznych (zadanie 8-10).

Zadanie 8-10. Przedstaw na ekranie „mapę fizyczną" funkcji dwóch zmiennych $f(x,y) = xsiny^2/(x+0,01)$ dla $0 \leq x \leq 10$, $0 \leq y \leq 4$, zaznaczając poszczególne przedziały wartości funkcji za pomocą różnych barw, np. $f(x,y) \leq -3$ — niebieski; $-3 < f(x,y) \leq -0,5$ — zielony; $-0.5 < f(x,y) \leq 0,5$ — żółty; $0,5 < f(x,y) \leq 3$ — pomarańczowy; $3 < f(x,y)$ — czerwony.

Zostałeś poproszony o pomoc w przygotowaniu programu służącego do przeprowadzania testów z różnych przedmiotów (zadanie 8-11). Mogliby z niego korzystać uczniowie przygotowując się do klasówek (lub egzaminów) z danego przedmiotu. Na początek chcesz zająć się prostszym koncepcyjnie programem do testowania znajomości tabliczki mnożenia, a później programem zawierającym kilkadziesiąt (lub więcej) pytań i po kilka odpowiedzi do każdego pytania (z których jedna jest poprawna). Ten drugi program mógłby być uniwersalny, umożliwiając odczytywanie plików z przygotowanymi oddzielnie pytaniami i odpowiedziami. Wówczas przygotowane pytania z odpowiedziami mógłbyś przechowywać w taki sposób, żeby przypadkowa osoba nie mogła ich odczytać (zadanie 8-12).

Zadanie 8-11. Przeprowadź test:

- z tabliczki mnożenia z losowo wybieranymi pytaniami (mechanizm losowy);

- wybierając losowo spośród przygotowanych pytań wraz z odpowiedziami do wyboru.
  Po każdej odpowiedzi ucznia potwierdzaj ją, jeżeli jest prawidłowa, a podawaj prawidłową, gdy jest błędna. Na koniec testu podawaj liczbę zadanych pytań i liczbę prawidłowych odpowiedzi.

Zadanie 8-12. Utwórz z danego pliku ASCII plik ze zmienionymi kodami niektórych (lub wszystkich) znaków.

Uwaga. Program taki może być przeznaczony do „zaszyfrowania" bądź „odszyfrowania" przygotowanych pytań i odpowiedzi testowych. Może też służyć do zamiany kodów polskich znaków.

Dane do programów użytkowych pochodzą niekiedy z innych programów, z urządzeń pomiarowych lub dokonujących odczytu dokumentów; bywają także otrzymywane za pośrednictwem sieci komputerowej (np. dane z katalogu bibliotecznego). Dane takie są często zapisane w pliku dyskowym. Rzadko kiedy plik taki można odczytać bezpośrednio w naszym programie bazy danych lub arkusza kalkulacyjnego, a w każdym razie efekt odczytania pliku może nam nie odpowiadać ze względu na inny układ danych, na zamieszczenie dodatkowych informacji, które nie są nam potrzebne, lub na użycie niewłaściwych separatorów oddzielających informacje od siebie. W takiej sytuacji jest niezbędne przekształcenie pliku z danymi (np. takie jak w zadaniu 8-13).

Zadanie 8-13. Odczytaj plik zawierający parzystą ilość liczb zapisanych po jednej w linii i przekształć go tworząc nowy plik (przeznaczony do wczytania do arkusza kalkulacyjnego), w którym liczby te byłyby zapisane po dwie obok siebie, oddzielone znakiem średnika.

Rozważaliśmy pokrótce (rozdz. 8.1.3) problem panowania nad archiwum zawierającym dużą liczbę dyskietek z różnymi plikami. Utworzenie odpowiedniej komputerowej bazy danych z pewnością ułatwiłoby wyszukiwanie odpowiedniej informacji. Problemem jest wprowadzanie informacji, i tu może być pomocny program napisany w języku uniwersalnym (zadanie 8-14). Wyszukanie informacji w bazie, np. określenie nazwy dyskietki, na której znajduje się szukany plik, może być z łatwością wykonane za pomocą poleceń systemu zarządzania bazą danych. Mogą jednak pojawić się pytania, na które znalezienie odpowiedzi nie jest takie proste, np. czy pliki znajdujące się na danej dyskietce są także na innych dyskietkach, lub ogólnie, które pliki są zapamiętane kilkakrotnie (zadanie 8-15). Wówczas można by skasować pliki zapisane na danej dyskietce i dyskietkę odzyskać. Jeżeli system zarządzania bazami danych umożliwia programowanie, to odpowiedni program mógłby być napisany w języku tego systemu.

Zadanie 8-14. Przekształć plik PLIKDIR utworzony poleceniem DIR /S > PLIKDIR wydanym w odniesieniu do napędu dysków elastycznych A (B) na plik ASCII, w którego każdym wierszu mieściłyby się informacje o poszczególnych plikach zapisanych na dyskietce znajdującej się w napędzie A (B): etykieta dysku (*volume*), nazwa pliku, rozszerzenie i jego rozmiar; dane powinny być zapisane w sposób umożliwiający import pliku do bazy danych (np. umieszczone w cudzysłowach i oddzielone od siebie przecinkami).

Zadanie 8-15. Przejrzyj tablicę z nazwami plików i wypisz tylko te nazwy, które występują w niej więcej niż jeden raz.

Przejdziemy teraz do następnego etapu, jakim jest tworzenie algorytmów i schematów działania programów. Po poznaniu sposobów przedstawiania algorytmów, skupimy uwagę w rozdz. 8.4.3. na tworzeniu algorytmów do (niektórych z) podanych zadań.

8.4. Algorytmy. Schematy działania programów

Algorytm określa „przepis" na zrealizowanie przez program postawionego zadania. Przepis ten ma postać ściśle określonych reguł i ma zapewniać zakończenie postępowania w skończonej liczbie kroków.

Znasz zapewne jakiś przepis kulinarny. Zapoznawszy się nim wiesz, jakie produkty powinieneś przygotować, żeby zabrać się za przyrządzenie potrawy lub napoju. Wiesz także, jakie czynności powinieneś wykonywać i w jakiej kolejności, a także kiedy (w jakiej sytuacji) należy je zakończyć.

Algorytmy służące do rozwiązywania problemów są przedstawiane z zachowaniem większej staranności co do formy algorytmu niż przy prezentowaniu przepisów kulinarnych. Do typowych form należy postać krokowa oraz sieć działań.

Nauka algorytmów może być wspomagana odpowiednimi programami (np. ELI). Umożliwiają one zarówno przedstawienie graficzne algorytmu (rys. 8.1), jak i demonstrowanie jego działania.

Przy przedstawianiu algorytmów będziemy używać zmiennych (takich jak w matematyce), oznaczanych symbolami złożonymi z jednej bądź wielu liter, i symboli podstawowych operacji matematycznych. Do umieszczenia danych i zmiennych będziemy korzystać (w razie potrzeby) z tablic podobnych do stosowanych w arkuszach kalkulacyjnych i bazach danych. Wykonywać będziemy także operacje na znakach i tekstach, podobne do znanych Ci z poprzednich rozdziałów.

Zmienną możesz sobie wyobrażać jako komórkę arkusza, tyle że opatrzoną **nazwą** (co w arkuszu jest też możliwe, ale nie jest konieczne).