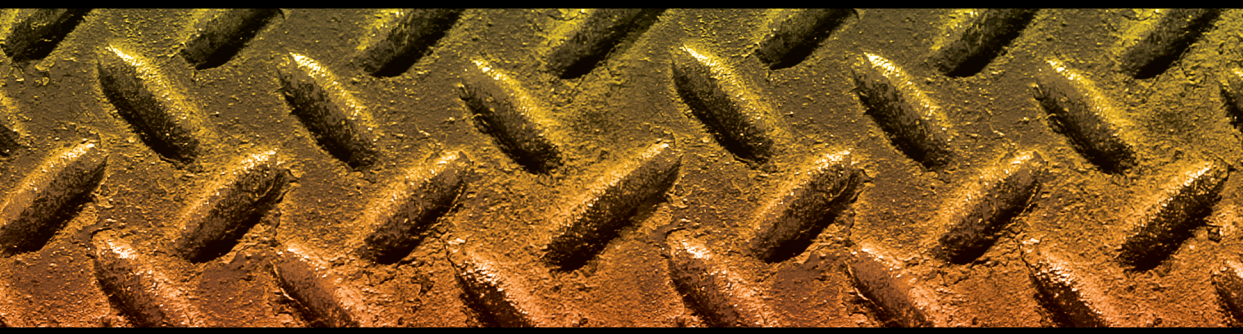
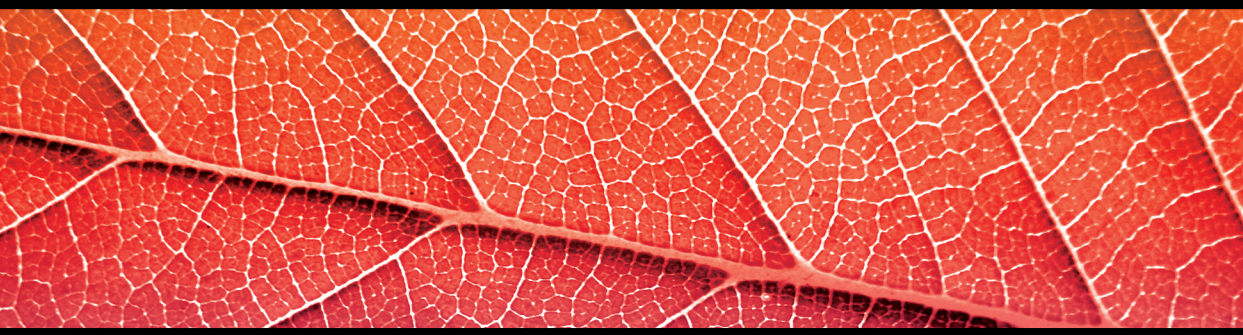
TOM 1

PROGRAMOWANIE
STRUKTURALNE

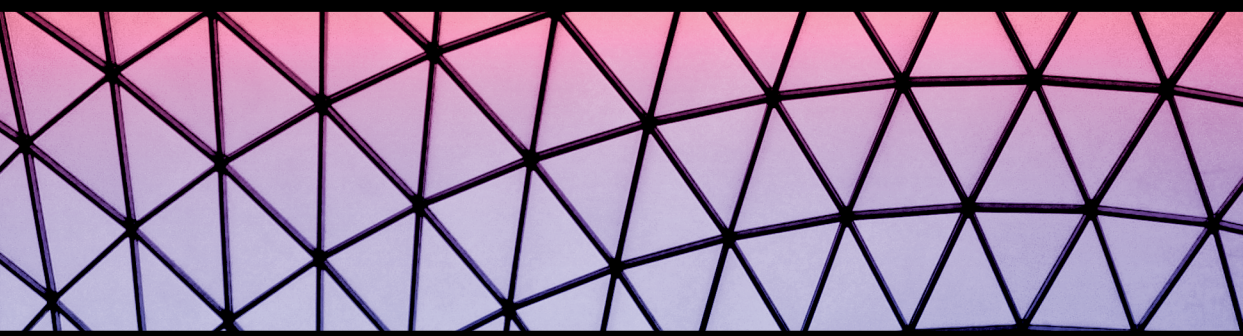
Krzysztof Wojtuszkiewicz



Programowanie



strukturalne i obiektowe



WYDAWNICTWO NAUKOWE PWN

KRZYSZTOF WOJTUSZKIEWICZ

PROGRAMOWANIE
STRUKTURALNE I OBIEKTOWE
Tom 1

WYDAWNICTWO NAUKOWE PWN
WARSZAWA 2009

Projekt okładki: **Łukasz Kowalik**

Redakcja: **Joanna Cierkońska**

Skład komputerowy: **Krzysztof Świstak**

Copyright © by Wydawnictwo Naukowe PWN SA
Warszawa 2009

Wszystkie prawa zastrzeżone. Reprodukcja bez zezwolenia zabroniona.

ISBN: 978-83-01-16042-5 t. 1
ISBN: 978-83-01-16043-2 t. 1-2

Wydawnictwo Naukowe PWN SA
02-676 Warszawa, ul. Postępu 18
tel. (0 22) 69 54 321
faks (0 22) 69 54 031
e-mail: pwn@pwn.com.pl
www.pwn.pl

Wydanie pierwsze
Arkuszy drukarskich 18
Druk ukończono w październiku 2009 r.
Druk i oprawa:
Drukarnia Wydawnictw Naukowych Sp. z o.o.
92-333 Łódź, ul. Wydawnicza 1/3

Spis treści

Wstęp

Książka **Programowanie strukturalne i obiektowe** jest przede wszystkim podręcznikiem do przedmiotu o tej samej nawie wykładanym dla zawodu technik informatyk. Może zainteresować jednak także tych wszystkich, którzy chcą poznać podstawy programowania i zrozumieć jego istotę.

Przedmiot **Programowanie strukturalne i obiektowe** jest przedmiotem obszernym. Obejmuje podstawy programowania i algorytmiki, języki programowania Pascal i C/C++ oraz wstęp do języka Java. Uczy zasad programowania strukturalnego i obiektowego, tworzenia programów z interfejsem tekstowym i graficznym, pracujących jako aplikacje konsoli (dawne aplikacje „dosowe") lub aplikacje do Windows. Tak szeroki zakres wymaganych zagadnień rodzi określone konsekwencje. Wiele problemów przedstawionych jest w książce skrótowo lub w sposób uproszczony. Wiele zagadnień musiałem pominąć, wybierając tylko te, które uznałem za istotne. Dlatego proszę o wybaczenie, jeżeli nie znajdziecie Państwo odpowiedzi na swoje określone problemy. Proszę jednak zdać sobie sprawę, że dogłębne przedstawienie choćby jednego języka programowania wymaga wielusetstronicowej publikacji (na przykład *Podstawy języka C/C++* Stanleya B. Lippmana i Josée Lajoie obejmują ponad 1200 stron). Zachęcam do szukania odpowiedzi na Państwa pytania w podanej literaturze. Jednocześnie będę niezmiernie wdzięczny za wszelkie uwagi i sugestie, co powinno się znaleźć w książce (a czego nie ma). Pozwoli to ulepszyć i lepiej dostosować następne wersje książki do Państwa wymagań. Uwagi proszę przesyłać na adres kawojt@ckp.pl.

Podręcznik będzie się składał z dwóch tomów. Tom pierwszy, który trzymacie Państwo w ręce, omawia podstawy programowania i algorytmów oraz programowanie strukturalne przy użyciu języków Pascal i C/C++. Tom drugi będzie obejmował programowanie obiektowe, programowanie do Windows, wybrane zagadnienia dotyczące algorytmów oraz wprowadzenie do programowania w języku Java.

Struktura pierwszego tomu książki jest następująca. Dwa pierwsze rozdziały: **Podstawy programowania** i **Algorytmy** pozwalają zrozumieć, na czym polega programowanie komputerowe. Opisują między innymi dwa ważne zagadnienia: cykl przygotowania programu oraz algorytmy. Rozdziały 3 i 4 dotyczą języka Pascal. Pierwszy przedstawia proponowane, bezpłatne środowisko (Free Pascal) do programowania w tym języku, a drugi obejmuje opis podstawowych struktur języka i ich zastosowań.

Rozdziały 5, 6 i 7 dotyczą programowania strukturalnego przy użyciu języka C/C++. W rozdziale 5 są omówione główne struktury tego języka oraz podstawy

obsługi środowiska C++ Builder. Rozdział 6 przedstawia operacje wejścia/wyjścia zarówno języka C jak i C++. Pierwsze warto znać choćby dlatego, że spotykamy jeszcze bardzo wiele programów napisanych w tym języku. Operacje wejścia/wyjścia dla C++ przedstawiane są z pewnym wyprzedzeniem i dokładne ich zrozumienie będzie możliwe po zapoznaniu się z programowaniem obiektowym (wówczas należałoby do tego rozdziału powrócić), jednak używamy ich już w programach tej części. Rozdział 7 dotyczy obsługi plików w języku C.

W rozdziale 8 piszemy o obsłudze debuggera w środowisku C++ Buildera. Podstawy uruchamiania programów podano w rozdziale 4. Temat ten jest często niedoceniany. Debugger to narzędzie niezbędne do uruchamiania programów, szczególnie większych, a jednocześnie doskonała pomoc dydaktyczna pozwalająca na śledzenie realizacji programu na różnych poziomach szczegółowości. Gorąco zachęcam do jego częstego użycia.

Pisząc niniejszą książkę, jako jeden z celów postawiłem sobie przygotowanie Czytelnika do zdania części pisemnej egzaminu potwierdzającego kwalifikacje w zawodzie technik informatyk. Około 30% pytań testowych tego egzaminu pochodzi z zakresu przedmiotu **programowanie strukturalne i obiektowe**. Pytania te sprawdzają podstawy znajomości algorytmów i języków programowania, takie jak przedstawianie algorytmów, składnia instrukcji języków Pascal i C/C++, działanie operatorów oraz stosowaną terminologię. Wszystkie te zagadnienia starałem się dokładnie w książce przedstawić. Na końcu pierwszego tomu, w dodatku B podaję przykłady zadań testowych. Pytania testowe związane z programowaniem strukturalnym i obiektowym ze wszystkich przeprowadzonych do tej pory egzaminów zawodowych zamieścimy na stronie internetowej PWN.

Na stronie tej zamieścimy też kody źródłowe wszystkich programów z książki, propozycje dodatkowych zadań oraz dodatkowe materiały związane z przedmiotem **programowanie strukturalne i obiektowe**.

Jako motto swojej książki o algorytmach Maciej M. Sysło umieścił zdanie wygłoszone przez Alberta Einsteina: „Things should be as simple as possible, but no simpler – Rzeczy powinny być tak proste, jak to możliwe, ale nie prostsze”. Pozostaje mi mieć nadzieję, że w książce udało mi się możliwie najpełniej tę prostotę uzyskać. Proszę jednak jeszcze raz o wszelkie uwagi, które pozwolą się zbliżyć do niej jeszcze bardziej.

Krzysztof Wojtuszkiewicz

Część pierwsza

Podstawy programowania

1. Podstawy programowania

1.1. Co to jest program komputerowy

Komputery biją obecnie wszelkie rekordy popularności. Ilościowo mogą z nimi konkurować chyba tylko telefony komórkowe, ale pod względem zakresu zastosowań i uniwersalności nie mają rywali. Spotykamy je w domu, w biurze, w małych i dużych firmach, w przemyśle, w instytucjach naukowych. Wykonują tyle zadań, że nie sposób wszystkich wymienić, ale trzeba wspomnieć o tym, o czym zwykle nie pamiętamy lub nad czym się nie zastanawiamy. Aby komputer wykonał jakiekolwiek zadanie, ktoś musiał napisać tak zwany program, który to umożliwi, lub nawet zestaw programów Jeżeli takim programem nie dysponujemy lub – co gorsza – nie da się go stworzyć (na przykład programu typującego ze stuprocentową trafnością wyniki totolotka), wówczas do realizacji takiego zadania nie możemy użyć komputera.

Czym więc jest program komputerowy? Najogólniej mówiąc, jest to sposób realizacji danego zadania, czyli zestaw kolejnych czynności, które ma wykonać komputer, a dokładniej jego procesor, przy czym ten „przepis" musi być zrozumiały dla układów komputera (przede wszystkim właśnie dla procesora) wykonujących poszczególne operacje. W dalszej części rozdziału wyjaśnimy to stwierdzenie, jednak bardzo zachęcamy do przestudiowania stosownych fragmentów rozdziału 3 książki *Urządzenia techniki komputerowej* [15], które znacznie dokładniej wyjaśnią działanie komputera na poziomie sprzętowym.

Stwierdzenie, że program komputerowy jest zestawem czynności umożliwiających realizację zadania, prowadzi do kwestii algorytmu, którą omawiamy nieco szerzej w podrozdziale 1.2 (wracamy do niej w kolejnych rozdziałach). Natomiast z warunkiem, że program musi być zrozumiały dla procesora komputera, wiążą się dwa zagadnienia: języków programowania (podrozdział 1.3) i sposobu realizowania programu przez komputer (podrozdział 1.4, przy czym ponownie polecamy podręcznik *Urządzenia techniki komputerowej*). Po tych wyjaśnieniach opisujemy cykl (sposób tworzenia) programu komputerowego (podrozdział 1.5).

1.2. Algorytmy

Jak powiedzieliśmy, zrealizowanie dowolnego zadania za pomocą klasycznego komputera wymaga przepisu postępowania, które prowadzi do realizacji tego zadania.

Przepis taki (przy określonych dodatkowych założeniach) nazywamy algorytmem. Zadania, które nie dają się zalgorytmizować, nie mogą być realizowane przez komputer.

Na potrzeby naszej książki przyjmiemy poniższą definicję algorytmu.

Definicja

Algorytmem nazywamy ściśle określony sposób postępowania prowadzący do rozwiązania określonej klasy zadań.

Nieco ściślejsza definicja, pochodząca z algorytmiki, czyli nauki o algorytmach, mówi, że algorytm to skończony ciąg dobrze zdefiniowanych operacji zastosowanych do skończonej liczby danych, prowadzący do rozwiązania określonej klasy zadań.

Robert Sedgewick w swojej książce *Algorytmy w C++* napisał: *algorytm jest metodą rozwiązywania zadań nadającą się do zastosowania w dowolnym komputerze i dowolnym języku programowania (...) algorytmy (...) obejmują metody organizacji danych biorących udział w obliczeniach. Obiekty powstałe w ten sposób nazywane są strukturami danych*. Poznając język programowania i jego możliwości, musimy poznać zarówno możliwości realizacji algorytmów, jak i tworzenia różnorodnych struktur danych.

Poniższy przykład prostego algorytmu rozwiązywania równania liniowego o postaci $ax + b = 0$ został przedstawiony w formie kolejnych kroków do wykonania. Zakładamy, że zadanie będzie rozwiązywane raczej na komputerze, a nie bezpośrednio przez człowieka. W przeciwnym razie na przykład ekran zastąpiłaby kartka.

1. Wczytaj wartości współczynników a i b.
2. Sprawdź, czy a jest różne od zera.
   2.1. Jeżeli TAK (czyli $a \neq 0$), to rozwiązaniem jest $x = -b/a$.
   2.2. Wyświetl x na ekranie.
   2.3. Koniec.
3. W przeciwnym wypadku ($a = 0$) sprawdź, czy b jest równe 0.
   3.1. Jeżeli $a = 0$ i $b = 0$, to równanie ma nieskończenie wiele rozwiązań.
   3.2. Wyświetl na ekranie komunikat „Równanie ma nieskończenie wiele rozwiązań”.
   3.3. Koniec.
4. W przeciwnym wypadku (czyli jeżeli $a = 0$ i $b \neq 0$) równanie nie ma rozwiązania (jest sprzeczne).
   4.1. Wyświetl na ekranie komunikat „Równanie jest sprzeczne”.
   4.2. Koniec.

O innych sposobach opisu algorytmów piszemy w podrozdziale 2.1.

1.3. Języki programowania

Chcąc stworzyć program komputerowy realizujący określone zadanie, trzeba zapisać algorytm rozwiązania tego typu zadania. Jednakże zapis zrozumiały dla procesora komputera jest dla człowieka nieczytelny. Dlatego opracowano narzędzia zwane językami programowania, pozwalające zapisywać w sposób czytelny dla człowieka kolejne polecenia dla komputera, tworzące program realizujący dane zadanie. Zapis taki nazywamy **programem źródłowym** lub **kodem źródłowym**. Musi on zostać przetłumaczony na postać akceptowaną przez procesor, który będzie ten program wykonywał. Postać tę nazywamy **programem docelowym** lub **kodem maszynowym**, a plik, w którym jest umieszczony, **plikiem wykonywalnym** (szerzej na ten temat piszemy w podrozdziale 1.5). Proces tłumaczenia nazywamy **kompilacją** albo **interpretacją** (są to dwie różne metody), a program, który tłumaczenie wykonuje, **kompilatorem** albo **interpreterem.**

Języki, w których zapisujemy programy, to:

- Język wewnętrzny procesora, zwany inaczej językiem maszynowym. Jest to jedyna postać programu akceptowana przez procesor. Poszczególne polecenia (operacje) dla procesora zapisywane są za pomocą ciągów zero-jedynkowych, będących ich kodami binarnymi (patrz listing 4). Przypominamy, że program zapisany w tym języku nazywamy programem docelowym lub kodem maszynowym.
- Języki symboliczne, zwane często językami wyższego poziomu, są zorientowane:
  - sprzętowo,
  - proceduralnie,
  - problemowo.

Języki zorientowane sprzętowo to przede wszystkim język asemblera (asembler). Jego cechą jest to, że jednej instrukcji napisanej w tym języku odpowiada jedna instrukcja maszynowa (jedna instrukcja w języku wewnętrznym procesora). Nazwy instrukcji w asemblerze są zwykle skrótami angielskich słów określających daną operację. Skróty te zwane są mnemonikami. Oczywiście instrukcje asemblera, jako instrukcje języka symbolicznego, muszą być przetłumaczone na język maszynowy. W tym wypadku proces tłumaczenia nazywa się **asemblacją**, a program tłumaczący to **asembler**.

Języki proceduralne zakładają zapis programu jako ciągu instrukcji do wykonania, przy czym zapis ten powinien być niezależny od procesora, który wykonuje program (wystarczy, by istniał do niego kompilator). Języki te są uniwersalne i można je podzielić na **strukturalne** oraz **obiektowe**. Pierwsze są przeznaczone, jak sama

nazwa wskazuje, do programowania strukturalnego, w którym program dzieli się na mniejsze fragmenty, czyli zadania łatwiejsze do zrealizowania (projektowanie top-down), i zapisuje w postaci podprogramów – procedur bądź funkcji. W programowaniu strukturalnym należy unikać instrukcji skoku (nawet jeżeli dany język ją oferuje), powodującej nieczytelność i zagmatwanie programu.

Języki obiektowe są kolejnym etapem rozwoju języków programowania. Po pierwsze, łączą dane z wykonywanymi na nich operacjami, tworząc tak zwane obiekty. Pozwala to na łatwiejsze i dokładniejsze modelowanie rzeczywistości. Po drugie, stosując takie techniki, jak dziedziczenie czy ukrywanie danych (enkapsulacja), ułatwiają realizację rozbudowanych programów przez duże zespoły programistyczne.

Językami strukturalnymi są na przykład Pascal oraz C, a obiektowymi – na przykład Java czy C++, przy czym ten drugi jest hybrydowy i umożliwia programowanie zarówno strukturalne, jak i obiektowe.

Języki zorientowane problemowo służą do programowania wysoko specjalizowanych systemów, takich jak roboty przemysłowe. Ich omówienie przekracza zakres niniejszego podręcznika.

Pokażemy teraz różnice między programem zapisanym w języku maszynowym i w językach wysokiego poziomu.

Rozpoczniemy od zapisu programu w językach wysokiego poziomu, Pascal i C++. Oto zapis algorytmu rozwiązywania równania liniowego z podrozdziału 1.2 w tych językach (listing 1 i 2). Cyframi pogrubionymi oznaczono instrukcje realizujące podstawowe kroki algorytmu (nie należy ich umieszczać w kodzie źródłowym programu). Zapis programów jest na razie niezrozumiały, ale nie należy się tym niepokoić – o zapisywaniu programów w Pascalu, C/C++ i Javie będzie mowa w dalszych częściach podręcznika, na razie wystarczy zwrócić uwagę na formę, stopień komplikacji i czytelność zapisu (znajomość angielskiego na poziomie podstawowym jest tu niezmiernie przydatna).

Listing 1. PP001

```
program Rown_lin;

var
  a,b : real;

begin
   Writeln('Podaj wartosc a');
1 Readln(a);
   Writeln('Podaj wartość b');
1 Readln(b);
```