Wstęp

Przystępując do projektowania systemu informatycznego, musimy dokładnie ustalić wymagania systemu oraz precyzyjnie określić realizowane przez niego zadania. Im bardziej skomplikowany jest system, tym bardziej szczegółowa powinna być specyfikacja zadań. W początkowym etapie projektowania systemu informatycznego bardziej nas interesuje, co program ma robić, a nie jak ma to robić. Na wstępie określamy zestaw danych wejściowych, wyjściowych oraz listę wykonywanych zadań. Projektowanie konkretnych struktur danych oraz implementację algorytmów powinniśmy odłożyć do późniejszego etapu prac, dzięki czemu możemy oddzielić wymagania i zadania systemu od konkretnej jego implementacji. Działanie każdego sytemu informatycznego opiera się na wielu warstwach abstrakcji. Określenie wymagań i zdefiniowanie zadań powinniśmy przeprowadzić dla każdej warstwy. Często mechanizmy abstrakcyjne wyższego poziomu budowane są na podstawie mechanizmów prostszych, określonych wcześniej na niższej warstwie. Narzędziem umożliwiającym definiowanie zadań i operacji wykonywanych na ustalonym zestawie danych jest *abstrakcyjny typ danych – ADT* (*ang*. *Abstract Data Type*). Abstrakcyjny typ danych opisuje zbiór wartości i operacji i jest dostępny jedynie za pośrednictwem interfejsu, który określa sposób dostępu do danych oraz wszystkie widoczne dla klientów *ADT* operacje. Dopiero gdy zostanie określona pełna jego specyfikacja, możemy przystąpić do implementacji, rozpoczynając od wyboru takiego sposobu organizacji danych, który jest najlepszy pod względem czasu wykonywania operacji i zajętości pamięci. Musimy pamiętać, że od prawidłowego wyboru odpowiednich struktur danych zależy efektywność algorytmów realizujących operacje *ADT*. Każdy abstrakcyjny typ danych może być zaimplementowany za pomocą różnorodnych struktur danych. Ich wybór w dużym stopniu zdeterminuje zastosowane algorytmy. Można powiedzieć, że struktury danych i algorytmy są w pewnym sensie budulcem abstrakcyjnego typu danych. Stąd też tytuł książki, który jest parafrazą klasycznej pozycji Niklausa Wirtha (*Algorytmy + Struktury Danych = Programy*).

Podstawowym tematem książki są abstrakcyjne typy danych oraz metody ich implementacji z wykorzystaniem różnorodnych struktur danych. Algorytmy omawiane są jedynie w kontekście struktur danych jako funkcje realizujące operacje interfejsu *ADT*. Szczególny nacisk położony jest na badanie i ocenę efektywności użytych struktur i algorytmów oraz na ich zastosowania praktyczne.

Książka powstała na podstawie materiałów i notatek do wykładu prowadzonego przeze mnie dla studentów Wydziału Matematyki i Nauk Informacyjnych Politechniki Warszawskiej. Jest adresowana do studentów informatyki oraz wszystkich osób zajmujących się zawodowo lub tylko zainteresowanych informatyką. Jej treść w dużym stopniu pokrywa się z programem przedmiotu *Algorytmy i Struktury Danych*. Przedmiot ten przez wiele lat prowadziliśmy wspólnie z dr. hab. Jerzym Wojciechowskim. Jego nagła śmierć w roku 2003 przerwała naszą współpracę. Wiele opisanych w książce rozwiązań i implementacji algorytmów jest efektem wspólnych przemyśleń i poszukiwań.

Materiał w tej książce przedstawiono w sposób dość ścisły i kompletny, choć nie nadmiernie sformalizowany. Czytelnik powinien dysponować jedynie podstawową wiedzą z matematyki na poziomie szkoły średniej. Omawiane metody i algorytmy ilustrowane są wieloma przykładami oraz rysunkami.

Zaprezentowane w książce algorytmy mają przeważnie postać funkcji języka programowania C++, rzadziej pseudokodu. Wybór C++ jako języka programowania podyktowany został wieloma względami. Język C++ jest szeroko wykorzystywany w konkretnych aplikacjach oraz równie szeroko w nauce i dydaktyce. Kod języka jest prosty i zrozumiały, a jednocześnie bardzo zwięzły. Zakładam, że czytelnik ma podstawowe przygotowanie z programowania. Starałem się, aby kod programów był bardzo czytelny, nawet kosztem wprowadzenia dodatkowych zmiennych czy instrukcji. Unikałem też mało czytelnych a często jedynie efektownych konstrukcji językowych. Stąd też implementacje algorytmów mogą być w prosty sposób przeniesione na inne języki programowania.

Książka składa się z 12 rozdziałów. Pierwsze dwa rozdziały poświęcone są zasadom projektowania i analizy algorytmów. Przedstawione są podstawowe metody oceny ich poprawności i efektywności oraz reguły ich projektowania. Zagadnienia te, a szczególnie pojęcie złożoności obliczeniowej używane są powszechnie w kolejnych rozdziałach do oceny prezentowanych algorytmów.

Rozdział 3 wprowadza i definiuje pojęcie struktury danych. Omawiane są tablice, listy oraz drzewa.

W kolejnych rozdziałach w sposób dość kompletny opisane są proste i złożone abstrakcyjne typy danych. Jest to najobszerniejsza część książki. Na początku przedstawione są *stosy* i *kolejki FIFO* oraz ich modyfikacje, takie jak *kolejki priorytetowe* oraz *kopce złączalne*. Najważniejszym abstrakcyjnym typem danych, przedstawionym w książce, jest *słownik*. Temu zagadnieniu jest poświęcony cały rozdział 7. Zawiera on szczegółowe omówienie drzew *BST*, drzew *AVL*, drzew samoorganizujących się *Splay*, *B-drzew* oraz *2-3* i *2-3-4* drzew w reprezentacji poziomo-pionowej i czerwono-czarnej. Przedstawiono również metody wyszukiwania danych oparte na wyszukiwaniu pozycyjnym oraz algorytmy *haszowania*. Rozdział 8 zawiera opis *kolejek konkatenowalnych*, które są prostym rozszerzeniem słowników.

Rozdział 9 jest poświęcony strukturom *Union-Find* oraz ich zastosowaniom. Szczegółowo omówiono tablicowe oraz drzewiaste ich realizacje.

Ostatnia część książki jest poświęcona algorytmom sortowania i wyboru. Ich znajomość jest niezbędna dla każdego, kto zajmuje się lub interesuje informatyką. Często wstępne uporządkowanie danych w znacznym stopniu upraszcza struktury danych wykorzystywane przy implementacji abstrakcyjnych typów danych.

Wszystkim, którzy przyczynili się do powstania tej książki, chciałbym złożyć serdeczne podziękowania. Nie sposób nie wspomnieć Jerzego Wojciechowskiego, którego bardzo mi brakowało przy jej końcowym redagowaniu. Szczególnie chciałbym podziękować studentom grup P1 i P2 Wydziału Matematyki i Nauk Informacyjnych Politechniki Warszawskiej, którzy cierpliwie czytając kolejne wersje tekstu, pomogli mi wychwycić błędy w programach i rysunkach oraz zasugerowali wiele usprawnień i poprawek.