МАТЕМАТИКА И МАТЕМАТИЧЕСКО ОБРАЗОВАНИЕ, 2006
MATHEMATICS AND EDUCATION IN MATHEMATICS, 2006
Proceedings of the Thirty Fifth Spring Conference of
the Union of Bulgarian Mathematicians
Borovets, April 5–8, 2006

МЕТОДИЧЕСКИ АСПЕКТИ НА ИЗУЧАВАНЕТО НА СТРУКТУРИ ОТ ДАННИ

Елка Тр. Иванова

В настоящата разработка се поставя въпросът за необходимо търсене на общата, езиково независима част в учебното съдържание по отношение на структурите от данни. Разгледани са някои основни понятия, заедно с езиково независима обща методика на изучаването им: като абстрактен тип данни, съответно структура от данни и т.н. Предложена е схема, която може да се следва при изучаването на основните абстрактни типове данни. Обсъдени са основните предимства на тази схема за ефективно обучение.

Изучаването на програмиране може да бъде разделено на три области: алгоритми, структури от данни и език за програмиране, с помощта на който се демонстрират основните методи и принципи. Най-често учебните пособия са посветени на конкретен език за програмиране, което поставя ударението върху изучаването на езика, като успоредно с това се разглеждат основни структури от данни и съответни алгоритми. Големият обем нова информация, свързана с езика за програмиране и разглеждането на конкретно представяне на основните структури от данни, налага извеждането от преподавателя на преден план на общите принципи и понятия. Необходимо е да се систематизират понятията в областта на структурите от данни и да се използва езиково независима методика на преподаването им. Направеното проучване през периода 2001 г.–2005 г. на студенти от втори и трети курс на специалност "Информатика" от Великотърновския университет, във връзка с трайността на усвояване на основни понятия от програмирането показва, че слабо място е усвояването на понятия като "абстракция на данни", "абстрактен тип данни" и "структури от данни". Приблизително 50% от анкетираните студенти (преди да са изучавали обектноориентирано програмиране) на въпроса "Що е структура от данни?" отговарят със списък от основни структури от данни или не дават отговор. Често отговорите им за дефиниция на основни понятия (като двоично дърво за търсене например) са конкретен пример за такова. Това проучване поставя необходимостта от построяване на езиково независима система от основни понятия, както и езиково независима методика на преподаването им. Още повече, че темата за структури от данни присъствува и в програмата на учениците от средното училище с профилирана подготовка по информатика. Настоящата работа разглежда някои методически аспекти на езиково независимата част от изучаването на структури от данни.

Основен метод на мислене в обучението по програмиране е абстрахирането и заедно с това – понятията "абстракция на данни" и "абстрактен тип данни" (АТД).

Разгледаният в [1] начин за дефиниране на понятието “АТД” позволява въвеждането му в началото на курс по програмиране (непосредствено след усвояването на “функции/процедури” и “контрол на типовете”), заедно с придружаващите го понятия “капсулиране”, “скриване на информация”, “интерфейс”, “програма-клиент” и т.н. Този начин на дефиниране се базира на принципа на абстракция. Чрез абстракция се създават добре дефинирани единици (величини), които могат да бъдат използвани по подходящ начин. Такива единици дефинират структура от данни от множество от един или повече елементи, която се нарича **абстрактна структура от данни (АСД)**. **АТД** се състои от АСД и съответни операции, наречени методи за достъп или интерфейс. АСД може да бъде достигната отвън само чрез методите на достъп. Абстрактният тип данни е тип данни, който осигурява **капсулиране** (дефиницията на типа данни и операциите върху обектите на типа са описани в една семантична единица) и **скриване на информация** (конкретните детайли по проектирането и представянето на обектите на типа са скрити от програмните единици – клиенти). Този начин на дефиниране на АТД има множество предимства:

1) Обучаваните ще усвоят техниката на абстракция още преди да развият навици за чисти програмистки техники. Ранното въвеждане на АТД създава навици за програмна реализация, които само трябва да бъдат допълвани и доразвивани при обектно ориентирания подход;

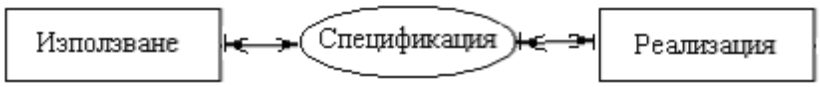
2) Дефиницията за **линейна АСД** може да бъде използвана за получаване на видовете АТД по отношение на линейност. Няма единна класификация на видовете структури от данни в езиците за програмиране, а и класификацията може да бъде направена по отношение на различни признаци. Най-често в учебниците структурите от данни се разделят на: прости и съставни; статични и динамични; линейни и нелинейни; циклични и нециклични СД. Да дефинираме линейна АСД: една АСД се нарича линейна, ако има следните две свойства: (1) Всеки елемент се следва от най-много един елемент; (2) Не съществуват два елемента, които да са следвани от един и същ елемент. **Примери:** Нека A, B, C, D, ... са елементи на АСД. Означението I -> J се използва в смисъл, че J следва I.

Масивът е линейна АСД: A->B->C->D

- Нарушение на (1): B<-A->C
- Нарушение на (2): A->C<-B

При нарушение само на свойство (1) се получава АСД “дърво” – няма два елемента, които да са следвани от един и същ. При нарушение и на (1), и на (2), се получава АСД “граф”. Една АСД се нарича циклична, ако съществува последователност от наследници, която довежда отново до първоначалния елемент (например A->B->C->D->A).

3) Дефинирането на АТД е езиково независимо (без частта за реализиране на АТД) и при това следва добре известните основни етапи на жизнения цикъл на програмни системи, показани на Фиг. 1:



Фиг. 1. Етапи на жизнения цикъл на програмни системи

По този начин се усвоява **основен подход за разработване и използване на програми** – реализацията е етап, независим от описанието; използването става чрез интерфейсна част, а то пък е независимо от реализацията.

4) За етапа "спецификация" (описание) може да се въведе езиково независим метод, като например предложеният в [2] формализиран, видоизменен за нуждите на обучение метод за описание на абстрактни типове данни, който се базира на т.нар. алгебрична спецификация и определя АТД като отношения между операции на типове. Ето модифицираният формат на алгебрично описание:

`АТД <име>`

`Използвани (други) АТД: ...`

`Неформално описание на АТД и неговите операции:`

Псевдокод, таблици, схеми и др.

`Операции:`

`Синтаксис:`

Име на всяка операция от АТД; типовете на параметрите и резултата

`Семантика:`

`Аксиоми (изисквания):`

Набор от аксиоми, които удовлетворяват операциите на АТД.

`Ограничения: ...`

Синтаксис: Означенията на операциите показват колко и какви са техните аргументи, и какъв е типът на формирания резултат. Например ако Int е тип "цяло число" и искаме да дефинираме синтактично операцията "събиране", то ще запишем: `Add: Int × Int -> Int`, където "×" е декартово произведение.

Семантика на АТД: Операциите изпълняват определени задачи. Семантиката включва аксиомите, които те удовлетворяват и евентуалните ограничения.

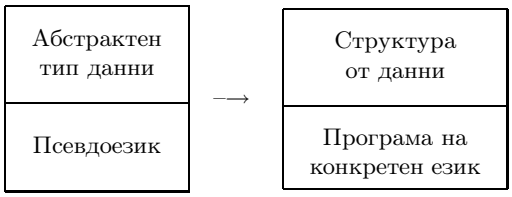
Описанието на АТД може да бъде изпълнено чрез средствата на един или друг език за програмиране (стига да разполага с такива). Такива средства са например интерфейси, сигнатури, абстрактни класове, абстрактни пакети и т.н., според терминологията на конкретния език. В такива случаи предварителното абстрактно описание се явява като условие на задача за реализация с конкретни средства. Ползата от въвеждане на формализирано описание е както от използването на математически модел - подчертават се съществените операции и техните свойства.

5) Описанието на основните операции на даден АТД изисква определяне на предварителни необходими условия (вход) и резултати от изпълнението им (изход). Създаването у обучаемите на навици за точно формулиране на "предусловия" и "следусловия" при проектиране на дадена операция (метод) намалява до минимум вероятността за допускане на логически грешки (които се откриват най-трудно) и е полезно при документирането на програми;

Езиково независимото дефиниране на АТД позволява **класифициране на основните операции** на АТД с цел акцентиране върху общото в тях. Възможни са различни класификации. Ето един пример:

- създаващи или генератори (... × ...-> T);
- запитващи (... × T × ...-> ...);
- изменящи (... × T × ...-> T).

7) На етап "Реализация" се дефинира по естествен начин понятието "структура от данни" (СД) като реализация на АТД (Фиг. 2) [1]:



Фиг. 2. Структура от данни е представянето на АТД
в конкретен език за програмиране

8) Дефинирането на понятието АСД позволява да се отстрани забелязана неточност в литературните източници по отношение на понятието "структура от данни". Например според [4, с.87] структура от данни е краткото наименование на структуриран тип данни на даден език за програмиране. Елемент на структуриран тип данни от своя страна представлява съвкупност от елементи, затова в спецификацията му трябва да се описват връзките между тях, както и допустимите операции на структурирания тип. Според учебник по информатика структурите от данни се класифицират като прости и съставни. Прости СД са числовите, логическите, знаковите, изброените, ограничените и др [3, с.198]. Очевидно би трябвало да се въведе единна терминология по отношение на СД (СД са структурирани типове според [4], а според [3] има прости СД). Това разминаване в терминологията се отстранява чрез дефинирането на АСД като елемент на АТД: абстрактната структура от данни може да бъде проста или съставна. **Проста АСД** е такава АСД, която е неразложима, т.е. не може да бъде разлагана повече на съставни части, т.е. няма съставни части, до които имаме индивидуален достъп (само една стойност може да се асоциира с променлива от този тип данни). Противоположна е ситуацията със **структурирана (съставна) АСД** – тя е разложима съвкупност от стойности, всяка от които може да бъде проста или структурирана. Една променлива от този тип се свързва не с една стойност, а с набор от отделни стойности, обединени в едно цяло. Например единичен символ от вградения тип **char** (в C или в Pascal) е неразложим, докато символният низ "Съставен" е композиран от 8 отделни символа. Простите АТД се превръщат в прости типове данни на ниво "език за програмиране", а структурираните АТД – в структурирани типове данни.

9) **Реализацията** (представянето) на АТД е свързано с подхода на програмиране, който се използва (изучава) и с конкретния използван език. Във връзка с етапа на представяне на АТД е целесъобразно разглеждането на общи начини за представяне на АТД. Познаването на основните механизми за формиране на физически структури в оперативната памет и съответните структури на езика за програмиране, които те представят, позволява да се разбере логиката на структуриране. По този начин могат да се въведат основните начини за разпределение на памет.

10) **Оптимизиране** на представянето на АТД. Тук се включват основните критерии за оптимизиране - различните начини на представяне се различават по скорост на обработване и сложност на съответния код; необходимо количество памет; брой на редовете на кода за представянето; възможността за лесна поддръжка на този код.

На ниво етап “Използване” на АТД се разглеждат примери за типични приложения на създавания АТД.

Разгледаната схема за дефиниране на АТД като общо понятие позволява еднотипно разглеждане на основните АТД и едновременно с това усвояване на важни основни понятия и техники, като по този начин спестява много учебно време и така допринася за ефективност на обучението.

ЛИТЕРАТУРА

[1] Е. Иванова. Един начин за въвеждане на понятията “абстрактен тип данни” и “структура от данни”, *Математика и математическо образование*, **33** (2004), 268–272.
[2] Е. Иванова. Описание на абстрактни типове данни в обучението по програмиране, *Математика и математическо образование*, **34** (2005), 327–332.
[3] П. Азълов, Ф. Златарова, М. Тодорова. Информатика за 10 клас. Профилирана подготовка”, Просвета, София, 2003.
[4] Т. М. Смит. Програмиране с Pascal, принципи и методи, Техника, София, 1996.

Елка Тр. Иванова
П.к. 7, ул “Н. Габровски “ 22
5000 Велико Търново, България
e-mail: elkaivanova@hotmail.com

ON THE METHODS OF TEACHING DATA STRUCTURES

Elka Tr. Ivanova

This paper puts for discussion the necessity of finding the common, language independent way of teaching data structures. In considers some base concepts and language independent methods of learning: abstract data type, data structures, etc. and also a common scheme for teaching all fundamental data structures. The base advantages of this way for effective teaching are considered.