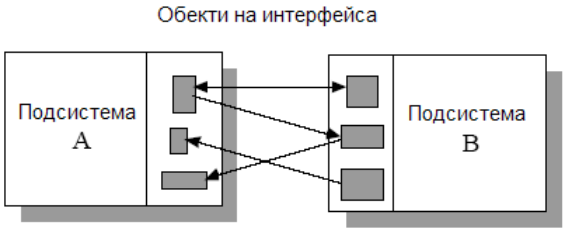
*МАТЕМАТИКА И МАТЕМАТИЧЕСКО ОБРАЗОВАНИЕ, 2005*
*MATHEMATICS AND EDUCATION IN MATHEMATICS, 2005*
*Proceedings of the Thirty Fourth Spring Conference of*
*the Union of Bulgarian Mathematicians*
*Borovets, April 6–9, 2005*

# ОПИСАНИЕ НА АБСТРАКТНИ ТИПОВЕ ДАННИ В ОБУЧЕНИЕТО ПО ПРОГРАМИРАНЕ

**Елка Т. Иванова**

С настоящата разработка се предлага един начин за въвеждане на понятието „абстрактен тип данни“ (АТД) още в самото начало на един курс по програмиране и по-конкретно, за използване на формални средства за неговото описание (модифициран алгебричен метод). Разгледани са примери. Начинът на задаване и описание на АТД е максимално езиково независим и унифициран, и следва етапите на разработване на програми въбще. Той е приложим за всички подход и на програмиране, включително за обектно-ориентирания. Чрез него обучаваните усвояват техниката на абстракция и умения за описание и използване на интерфейс на АТД.

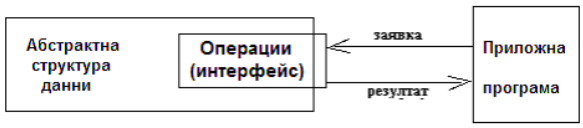
В съвременното програмиране, за да бъдат създадени, големите програмни системи се разделят на подсистеми, които се реализират независимо. Подсистемите евентуално използват други подсистеми и затова съществена част от процеса на определянето им (специфицирането им) е да се дефинира техният интерфейс. Той представлява множество от абстрактни типове данни (АТД) и обекти:




Всяка подсистема реализира тези интерфейси и те са достъпни за цялата система. Затова е особено важно интерфейсът да е ясно определен. Както се вижда, описанието на АТД като елемент на този интерфейс, както и реализирането им, е важна част от създаването на програмната система като цяло. При изучаване на информатика и по-конкретно, на програмиране и структури от данни, успоредно с усвояването на конкретен език за програмиране, особено важно е усвояване на основни принципи и методи. Един от най-важните за бъдещите програмисти, а и за бъдещите учители по информатика, е принципът на абстракция и в частност, на абстракция на данни. Основно понятие, тясно свързано с абстракцията на данни,

е понятието „абстрактен тип данни“. Един начин за ранното въвеждане на това понятие и свързаните с него „капсулиране“, „скриване на информация“ е това да бъде направено в самото начало на обучението по програмиране, например заедно с понятието „модул“ и непосредствено след усвояването на „функции/процедури“ и „контрол на типовете“ [1]. Предимство на такова ранно въвеждане е унифициране на разглеждането на АТД (съответно типове данни) без значение какъв подход се използва за тяхното представяне. Обучаваните усвояват техниките на абстракция, които само трябва да бъдат допълвани и доразвивани при обектно-ориентирания подход. Понятието „структура от данни“ се въвежда по естествен начин като реализация на АТД в средата на конкретен език за програмиране. Разглежданият начин на дефиниране и създаване на АТД следва етапите на разработване на програмни системи въобще, а именно: използване, спецификация (описание), представяне (реализация). Само етапът на представане (реализация) на АТД е езиково зависим и при него може да се използват различни подходи според нивото и целта на обучение.

Ето кратко описание на този начин за въвеждане на понятието „АТД“: АТД е единица, която се състои от абстрактна структура от данни (АСД) и множество от операции, наречени интерфейс. Структурата от данни може да бъде достигната само чрез дефинираните операции (методи за достъп) (Фиг. 1).




*Фиг. 1 Абстрактен тип данни*

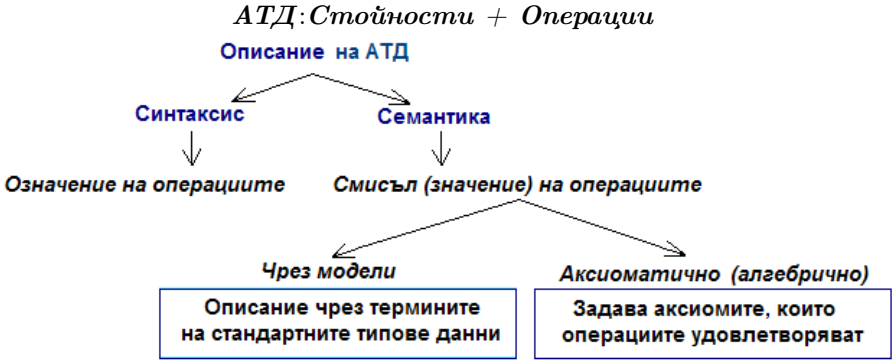
Обект на настоящо разглеждане са начините на описание (специфициране) на АТД. Представянето на АТД в конкретен език за програмиране го превръща в структура от данни. Използването му, благодарение на дефинирания интерфейс, е независимо от реализацията му. Тъй като основна задача е усвояването на общи принципи в процеса на конкретно изучаване на даден език за програмиране, тук е дадено предимство на **езиково независимото описание на АТД.** Целта е в процеса на обучение да се създадат навици за точно и ясно описание на интерфейса на АТД чрез формални средства.

**Описание на АТД.** Известно е, че общото **описание на даден АТД** се извършва с помощта на езици за описание на АТД, които биват естествени езици, формални езици и др. Най-често в учебните пособия се използва неформален език от вида:

***Псевдокод = Синтаксис на конкретен език за програмиране + Естествен език***

по подобие на описанието на алгоритми. Поради огромния обем информация при изучаване на информатика, целесъобразно е ранното въвеждане на формално унифицирано описание на АТД, разбира се, придружено от неформални описания с псевдокод и/или таблици и схеми. Езиково независимото описание на АТД трябва да включва: име на АТД, други използвани АТД, дефиниране на множеството от допустими стойности, дефиниране на допустимите операции. От своя страна, дефи-

нирането на множеството от операции включва: синтактична спецификация (формата), семантична спецификация (значението) и ограничения. Съществуват няколко начина за задаване на семантиката на операциите: текстово описание на свойствата; чрез модели; чрез алгебрична спецификация. Тук се дава предпочитание на алгебричното (аксиоматично) описание, понеже то задава АТД чрез операциите и отношенията между тях. Алгебричното описание обхваща най-общите свойства на всички възможни представяния.




**Формат на алгебричното описание.** Уместно е общият формат за задаване на алгебрично описание да се опрости за нуждите на обучението, като се пропуснат понятия, свързани с него като „алгебра“„сорт“ и др. Ето един опростен общ вид на алгебрично описание на АТД:

***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:***

Набор от аксиоми, които удовлетворяват операциите на АТД.

**Ограничения:...**

**Синтаксис на АТД:** Един АТД се определя синтактично от името си и от означенията на множеството от основни операции. 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 x Int -> Int[2]`.

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

**Пример 1:** Да разгледаме описанието на АТД „Стек“. За целите на обучение неформалното описание е изтеглено преди формалното:

**Логическо (неформално) описание:** *Абстрактна структура от данни със свойството, че е линейна последователност, за която операциите включване и изключване на елемент са допустими само от единия ѝ край.* Уместно е дефиницията да се подкрепи със схема (не е зададена поради ограничение на обема) или още по-добре аплет. Неформалното описание на основните операции, заедно с предусловията и следусловията, може да се представи, например, таблично:

| | |
|---|---|
| **CREATE()** | Създаване на празен стек<br>• Вход: няма Предсловие: няма<br>• Изход: S (стек) Следусловие: S е създаден и празен |
| | Поставяне на X в стека S, така че Top(Push(X,S))=X.<br>• Вход: X (елемент) и S (стек)<br>• Изход: S′ (т.е. S е променен)<br>• Предсловие: X е от тип като елементите на S<br>• Следусловие: S′ има за елемент на върха си X и за останалите елементи – S. |
| **POP(X, S)** | Извличане на елемент от стека S (който е бил въведен в S последен) и той става стойност на X<br>• Вход: S (стек)<br>• Изход: S′ (т.е. S е променен)<br>• Предсловие: S не е празен<br>• Следусловие: Понеже S не е празен, той се състои от 2 части: елемент на върха T и стек R от останалите елементи. S′ = R. |
| **IsEMPTY(S)** | Връща булева стойност 'True', ако стекът S е празен; 'False' – ако не.<br>• Вход: S (стек)<br>• Изход: IsEmpty (boolean)<br>• Предсловие: –<br>• Следусловие: IsEempty има ст. True, ако S е празен. |
| **TOP(S)** | Достъп до елемента на върха на стека S. Този елемент не се извлича от стека. Еквивалентна е на: POP(X,S), достъп до елемента, PUSH(X,S)<br>• Вход: S (стек) Предусл.: S не е празен<br>• Изход: E (ел. на S) Следусл.: E е ел. от върха на S (S се запазва) |

**Формално алгебрично описание:**
**АТД <Stack>**
***Използвани АТД:*** Boolean, Element
***Операции:***
***Синтактична спецификация:***
- *create*: → Stack
- *push*: Element × Stack → Stack
- *top*: Stack → Element
- *pop*: Stack × Element → Stack
- *isEmpty*: Stack → Boolean

***Семантична спецификация (аксиоми):***

1. $isEmpty(create) = true$
2. За всяко $x, s$ $isEmpty(push(x, s)) = false$
3. За всяко $x, s$ $top(push(x, s)) = x$
4. За всяко $x, s$ $pop(push(x, s)) = (x, s)$

***Изключения:***

- $top(create)$ е некоректно
- $pop(create)$ е некоректно

**Пример 2:** Описание на АТД „SET“ от целочислени елементи. Разгледано е само описанието на основните операции на АТД.

**АТД < SET>**
***Използвани АТД:*** Boolean, Element
**Операции:**
***Синтактична спецификация:***

*newset*: → Set
*add* : Set × Integer → Set
*delete*: Set × Integer → Set
*member*: Set × Integer → Boolean
*empty* : Set → Boolean
*size* : Set → Integer

***Семантична спецификация (аксиоми):***

delete(newset,I) = newset
delete(add(S,I),J) = S if I=J else add(delete(S,J),I)
member(newset,I) = false
member(add(S,I),J) = true if I=J else member(S,J)
empty(newset) = true
empty(add(S,I)) = false
size(newset) = 0
size(add(S,I)) = size(S)+1 if I is not in S else size(S)

**Изключения:** Няма.

Описването (задаването) 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 ранното въвеждане на абстрактни типове данни и на тяхното абстрактно описание в началото на обучението по програмиране е трайно овладяване на техниките на абстракция. Така уменията на обучаваните само ще бъдат допълвани в следващи специализирани курсове и в бъдещата им практическа работа.

## 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] N. Dale, D. Teague. C++ Data Structures. Jones and Bartlett Publishers, Massachusets, 2001.

ПФ на ВТУ „Св. Св. Кирил и Методий“
5000 Велико Търново
e-mail: elkaivanova@hotmail.com

# SPECIFICATION OF ABSTRACT DATA TYPES IN TEACHING PROGRAMMING

**Elka T. Ivanova**

This paper proposes a one way of defining "'Abstract Data Type"' (ADT) early in the beginning of the first programming course and in particular the formal specification of ADT (modified algebraic specification). Some examples are considered. The way of specification is language independent and unified, and it follows the steps of creating application software. It is useful for all programming methods, including the object-oriented one. It helps students to learn abstraction techniques and ability to specify and use interfaces of ADT.