МАТЕМАТИКА И МАТЕМАТИЧЕСКО ОБРАЗОВАНИЕ, 2004
MATHEMATICS AND EDUCATION IN MATHEMATICS, 2004
Proceedings of the Thirty Third Spring Conference of
the Union of Bulgarian Mathematicians
Borovets, April 1–4, 2004

ЕДИН НАЧИН ЗА ВЪВЕЖДАНЕ НА ПОНЯТИЯТА „АБСТРАКТЕН ТИП ДАННИ“ И „СТРУКТУРА ОТ ДАННИ“

Елка Т. Иванова

С настоящата разработка се предлага един начин за въвеждане на понятието „абстрактен тип данни“ още в самото начало на един курс по програмиране. Той е максимално езиково независим и следва етапите на разработване на програми въобще. Понятието „структура от данни“ е въведено като конкретно реализиране на абстрактен тип данни със средствата на езика за програмиране. Предложеният начин е приложим за всички подходи на програмиране, включително за обектно ориентиране. Чрез него обучаваните усвояват техниката на абстракция (включително капсулиране и скриване на информацията) преди да се развият навици за чисти програмистки техники.

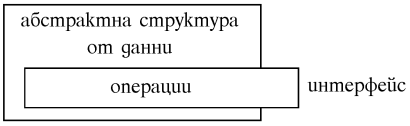
В процеса на програмиране понятията „абстрактен тип данни“ (АТД) и „структура от данни“ (СД) заемат съществено място. Обучението по програмиране обикновено се разделя на две части: първа част, при която се използват структурният и модулният подходи на програмиране и втора част – обектно ориентирано програмиране. Съществуват различни методики за въвеждане на понятието АТД, а именно: не се използва в първата част и се въвежда във втората; дефинира се при модулно програмиране; дефинира се още при структурно програмиране. Имайки предвид важността на обектно ориентирания подход за реални разработки, има смисъл понятието АТД и свързаните с него „абстракция“, „капсулиране“, „скриване на информация“ да се разгледат в самото начало на обучението по програмиране (това би отнело около един учебен час) в същия вид, както в обектно ориентираната част. Например това би могло да стане заедно с понятието „модул“ непосредствено след усвояването на „функции/процедури“ и „контрол на типовете“. Така се осигуряват и средства за представяне на АТД. По този начин обучаваните ще усвоят техниката на абстракция преди да развият навици за чисти програмистки техники. Ранното въвеждане на АТД и модули създава навици за програмна реализация, които само трябва да бъдат допълвани и доразвивани при обектно ориентирания подход. Настоящата разработка предлага един примерен начин за въвеждане на понятията АТД и СД, който е езиково независим (без частта за реализиране на АТД). При това той следва основните етапи на разработване на програми, а именно – анализ и проектиране, реализация, използване и съпровождане. Направените експерименти с него показват повишена ефективност на усвояване. Изложеното тук би могло да

бъде полезно както за студенти, така и за ученици в дисциплината информатика, в модулите за структури от данни и програмиране.

I. Абстракция и проектиране на АТД. Чрез процес на моделиране (**абстракция**) се създава модел на даден реален проблем. Създаденият модел дефинира една абстрактна представа на проблема, която е фокусирала само неговите съществени свойства. Тези свойства включват:

- данните, които са съществени;
- операциите, които са дефинирани от проблема.

И така, абстракцията е структуриране на проблема чрез дефиниране и комбиниране на подходящи данни и операции, като все още не се интересуваме от това как ще бъдат представени данните и реализирани операциите (скриване на детайлите). Чрез абстракция се създава добре дефинирана единица (величина), която може да бъде използвана по подходящ начин. Такива единици дефинират структура от данни от множество от един или повече елементи, която се нарича **абстрактна структура от данни (АСД)**. АСД може да бъде достигната само чрез съответни дефинирани операции - само операциите са видими отвън. Множеството от тези операции се нарича интерфейс или методи за достъп. Единица с току-що изброените свойства се нарича **абстрактен тип данни (АТД)** (Фиг. 1).



Фиг. 1. Абстрактен тип данни (АТД)

Абстрактният тип данни е тип данни, който осигурява:

1) **Капсулиране** (обединяване) – Представянето или дефиницията на типа данни и операциите върху обектите на типа са описани в една семантична единица. Това позволява АТД да бъде включван като елемент на друг АТД.

2) **Скриване на информация** – Конкретните детайли по проектирането и представянето на обектите на типа са скрити от програмните единици, които използват типа. Това означава, че единствените операции, които могат да бъдат приложени върху обектите на типа, са тези, които са предоставени при дефинирането на типа. Програмните единици, които използуват даден АТД, се наричат „**клиенти**“.

Предимствата на скриването на информация са:

- Детайлите по реализацията са скрити, така че кодът „клиент“ не зависи от конкретна реализация (тя може да бъде променяна независимо);
- Кодът-клиент е по-надежден, т.е. той не може, дори и случайно да промени реализацията на обектите на АТД, което пък подобрява неговата цялостност.

Една формална спецификация не е завършена, ако не са определени точно условията на нейния вход („**предусловие**“) и изход („**следусловие**“). Създаването на навици за точно формулиране на входните и изходни условия при проектиране

на дадена операция (метод) намалява до минимум вероятността за допускане на логически грешки и е полезно при документирането на програми.

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 се използва неформален език от вида:

Псевдокод = Синтаксис на конкретен език за програмиране + Естествен език по подобие на описанието на алгоритми. Езиково независимото описание на АТД би трябвало да включва: **синтактична спецификация** (формата); **семантична спецификация** (значението); **ограничения**.

Ето един пример за езиково независимото описание на операциите на АТД „опашка“:

Синтактична спецификация:

NewQ( ) → queue

AddQ(queue, item) → queue

DelQ(queue) → queue

FrontQ(queue) → item

IsEmpty(queue) → queue

Семантична спецификация:

IsEmpty(NewQ( )) → true

IsEmpty(AddQ(Q, I)) →false

DelQ(AddQ(Q,I)) → **if** IsNewQ(Q) **then** NewQ()
else AddQ(DelQ(Q,I))

Ограничения:

FrontQ(NewQ()) → Error
DelQ(NewQ()) → Error

Основните операции с даден АТД могат да бъдат класифицирани с цел акцентиране върху общото в операциите с отделните АТД. Например в [2] авторите класифицират основните операции на произволен АТД. С оглед на подготовка за използване на обектно ориентиран подход една подходяща класификация е следната. Нека Ф е множеството на всички елементи (обекти) на даден АТД, означен с Т. **Основните операции** на АТД могат да бъдат класифицирани в следните групи:

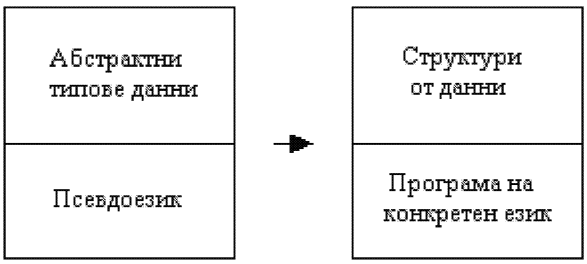
- *Конструктори* – операции, с които се конструират (създават) елементи на Ф;
- *Деструктори* – операции за унищожаване на вече създадена АСД;
- *Селектори* – чрез тях се извършва достъп до компоненти от типа Т, които са градивни компоненти на елементите на Ф; достъпът до елемент изисква определянето на позицията на елемента в АСД;
- *Предикати* – операции, с които се извършват сравнения на елементи от множеството Ф, проверка за празнота на АСД, за принадлежност към АСД;
- *Операции с общо предназначение* – това са обединение и разделяне на АСД, копиране на стойност на АТД и т.н.

II. Реализация. При представяне на даден АТД в конкретен език за програмиране програмистът трябва да:

- избере конкретно представяне на абстрактната структура данни на АТД, използвайки типовете данни, поддържани от конкретния ЕП;
- програмира всяка допустима операция на конкретния ЕП.

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.

Представянето на даден абстрактен тип данни в конкретен език за програмиране, го превръща в **структура от данни**. Този процес на представяне (реализиране) на АТД може да се илюстрира нагледно чрез следната схема:

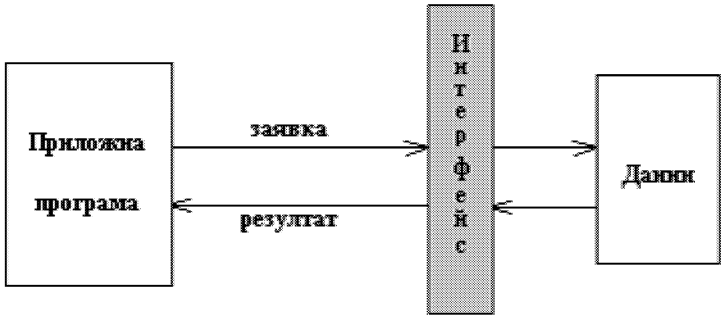


Реализацията (представянето) на АТД е свързано с подхода на програмиране, който се използва. При структурния подход то става чрез използване на подходящи декларации на типове и функции или процедури за реализиране на основните операции. Модулният подход предлага самостоятелна програмна единица – модул, обединяваща в едно цяло структурата данни и нейните основни операции. Принципът на абстрактност е изцяло в основата на обектно ориентираното програмиране. При него класовете са много удобно средство за представяне на АТД (те представляват представяне на потребителски дефиниран АТД) и те могат да бъдат разгледани като наследници на модулите в структурното програмиране. Класовете обединяват данните и операциите в едно цяло. Конкретен представител на даден клас се нарича обект, а функциите, реализиращи операциите на АТД се наричат методи.

III. Използване на АТД. Използването на създадена структура от данни (АТД) не изисква познаването на детайлите по реализацията ѝ. Интерфейсът за достъп до данните на структурата са основните операции на дефинирания АТД. Това се вижда добре от следната схема:

Вградените типове данни в един език за програмиране са конкретна реализация на съответен АТД, която е осъществена от разработчиците на езика [1].

Навременното въвеждане на основното понятие АТД и систематичното му използване, освен че изгражда добър стил на програмиране, очевидно би улеснило чрез съответна аналогия преминаването към обектно ориентирания подход на програмиране. Спазването на обща схема при разработване на програми и в частност на АТД преодолява в известна степен противоречието между непрекъснато увеличаващия се обем информация и възможностите на обучаваните за нейното консумиране,



Програмистът на приложната програма иска да създаде или използва обекти на даден АТД без да се интересува от детайлите по реализацията.

Програмистът, който реализира даден АТД се съсредоточава върху детайлите по представянето му в конкретния език за програмиране без да се интересува от приложните програми, които ще използват получената структура от данни.

като по този начин допринася за увеличаване на ефективността на усвояване на учебния материал.

ЛИТЕРАТУРА

[1] Т. Смит. Програмиране с Pascal – принципи и методи, София, Техника, 1996.
[2] N. Dale, C. Weems, M. Headington. Programming and Problem Solving with C++, Massachusets, Jones and Bartlett Publishers, 2000.

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

ONE WAY FOR INTRODUCING ABSTRACT DATA TYPES AND DATA STRUCTURES

Elka T. Ivanova

The paper considers a way of defining “Abstract Data Type” (ADT) early at the beginning of one programming course. It is independent on the programming language used and it follows the program development process. A data structure is defined as a concrete implementation of an ADT with a given programming language. This way of introducing ADT is good for structural design as well as for object-oriented design. It helps students to learn the abstraction technique (including encapsulation and information hiding) before developing abilities for using programming techniques.