ОБЕКТНООРИЕНТИРАНИЯТ ПОДХОД ПРИ ИЗУЧАВАНЕТО НА СТРУКТУРИ ОТ ДАННИ

Елка Трифонова

Информатиката е наука, която се развива с невероятно бързи темпове, особено през последните години. Това развитие налага много бърза промяна на учебното съдържание на всички дисциплини, свързани с нея и нейното прилагане. Това особено силно се отнася за програмирането и в частност, за структурите от данни. Известно е, че структурите от данни представляват конкретна езикова реализация на съответни абстрактни типове данни, от което следва, че развитието на езиците за програмиране се отразява и върху начините за тяхното представяне. Следването на създадени учебни традиции (ако може да се говори за традиции в рамките на две или три десетилетия) е задължително. Но в същото време е наложително и разглеждането на структурите от данни в нова светлина – а именно от гледна точка на обектно-ориентирания подход. Обектноориентираното програмиране (ООП) предлага нов и мощен модел за създаване на софтуер, който ускорява разработването на програми, подобрява поддържането, актуализирането и многократното им използване. Това обяснява популярността на обектно-ориентирани езици за програмиране като Java, C++, Smalltalk и т.н. и изисква студентите по информатика да напуснат университета с формирани умения за работа в обектноориентирана среда. Да разгледаме един пример за представянето на абстрактния тип данни стек в структурата стек с популярния за обучение език Паскал (в последните си версии предлага обектноориентирани средства) чрез обектноориентиран подход (вляво на Фиг. 1) и чрез модулния (вдясно на Фиг. 1):

Както се вижда от примера, разликата в представянето е минимална с огромни предимства за обектния подход, от което следва, че не би било трудно за обучаваните да получат навици за използване на обектния подход на по-ранен етап. Използването на модулния подход в начален курс по програмиране подготвя студентите за бързо преминаване към ООП, а ако курсът по структури от данни следва този по ООП (все още го предшествува в

```
unit StackOBJ;
interface
type  Pointer = ^Node;
       Node = record
                  Data : Char;
                  Next : Pointer
              end;
 Stack = object
    constructor Init;
    function Empty : Boolean; virtual;
    procedure Push ( Item : Char );  virtual;
    procedure Pop; virtual;
    procedure GetTop ( var Item : Char );
                                        virtual;
     private
        Stack : Pointer;
     end;
implementation
  constructor Stack.Init;
       begin    Stack := nil     end;

  function Stack.Empty : Boolean;
      begin     Empty := Stack = nil   end;

  procedure Stack.Push ( Item : Char );
  var
    Temp : Pointer;
  begin
    New( Temp );
    Temp^.Data := Item;
    Temp^.Next := Stack;
    Stack := Temp;
  end;

  procedure Stack.Pop;
  var   Temp : Pointer;
  begin
    Temp := Stack;
    Stack := Stack^.Next;
    Dispose( Temp )
  end;

  procedure Stack.GetTop ( var Item : Char );
  begin
    Item := Stack^.Data
  end;

end.
```

```
unit Stack;
 interface
 type  Pointer = ^Node;

       Node = record
                 Data : Char;
                   Next : Pointer
               end;

 procedure Init (var Stack:Pointer);
 function Empty (var Stack:Pointer):    Boolean;
 procedure Push ( Item : Char;var stack:Pointer );
 procedure Pop(var Item:Char;var stack:Pointer);
 procedure GetTop (var Item:Char; stack:Pointer);


 implementation
 procedure Init(var stack:Pointer);
  begin    Stack := nil     end;

 function Empty( stack:Pointer): Boolean;
  begin     Empty := Stack = nil   end;

 procedure Push ( Item : Char;var stack:Pointer);
   var
     Temp : Pointer;
   begin
     New( Temp );
     Temp^.Data := Item;
     Temp^.Next := Stack;
     Stack := Temp;
   end;

 procedure Pop(var Item:Char;var stack:Pointer);
   var
     Temp : Pointer;
   begin
     Item:=Stack^.Data;
     Temp := Stack;
     Stack := Stack^.Next;
     Dispose( Temp )
   end;

 procedure GetTop (var Item : Char;stack:Pointer);
   begin
     Item := Stack^.Data
   end;

 end.
```

Фиг. 1

учебните програми), то те ще имат възможност за създаване на трайни умения за работа в обектно-ориентирана среда и ще бъдат добре подготвени за съвременните изисквания за работа като програмисти. Някои специалисти дори отиват по-далеч, като твърдят, че процедурният подход не бива да предшествува обектния, ако крайната цел е обектно-ориентираният подход (1). А нека се има предвид, че за разработване на софтуер в последните години се използват именно обектно-ориентирани езици. Последователността на изучаване на дисциплините, разглеждащи структури от данни е "Програмиране" и "Структури и данни" (например така е във ВТУ в настоящия момент), тези учебни дисциплини използват структурния и модулния подход и предшествуват курсовете по обектноориентирано програмиране. Целта на този доклад е да постави на вниманието на преподаватели във ВУЗ, а също така и на учителите по информатика, занимаващи се със свободноизбираема подготовка и подготовка за олимпиади, необходимостта от предварително създаване на умения у обучаваните за използване на модули в часовете по програмиране, което гарантира лесното преминаване към ООП, както и търсене на варианти за ранно изучаване на ООП. Една възможност е след изучаване на масиви да бъдат разгледани елементи на ООП, както и понятието "абстрактен тип данни", след което представянето на основните структури от данни да става с модулно и ООП. Някои големи университети в света вече са преусроили своите курсоове по структури от данни към обектноориентирания подход, като например университета на щата Колорадо, САЩ (www.cs.colostate.edu/~cs200), университета в Уисконсин, САЩ, (www.uwplatt.edu) и други. Такава промяна на учебното съдържание би била допълнителна мотивация за студентите, че се обучават в среда, близка до реалната и би подобрила качеството на обучение на бъдещите специалисти по информатика.

ЛИТЕРАТУРА

1. Why Procedural is the Wrong First Paradigm if OOP is the Goal", Joseph Bergin, Pace University, www.csis.pace.edu/ ~ bergin/papers/whynotproceduralfirst.html.

THE OBJECT-ORIENTED APPROACH TO DATA STRUCTURES LEARNING

ELKA TRIFONOVA

Summary

This report considers the need of learning data structures with object oriented programming. This is because students must be prepared for real work in object oriented environment — the most popular environment in the software market today.