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

ВЪВЕДЕНИЕ В ОБЕКТНООРИЕНТИРАНОТО ПРОГРАМИРАНЕ С OBJECT PASCAL И DELPHI

Христо Д. Крушков, Мариана Ив. Крушкова

Обучението по програмиране в средното образование е свързано с три езика за програмиране – Бейсик, Паскал и Си. В задължителната подготовка не е предвидено задълбочено усвояване на работата с обекти и обектно-ориентираното програмиране. Този пропуск в обучението на по-талантливите ученици се компенсира в часовете по СИП. В настоящия доклад е предложен модел за въведение в обектно-ориентираното програмиране посредством Object Pascal и визуалната среда за програмиране Delphi, който може се използва в тези часове.

Изборът на обектно-ориентираното разширение на езика Паскал за въведение в обектно-ориентираното програмиране (ООП) се базира на следните доводи:

- налице са основните концепции на ООП;
- възможностите за работа с класове и обекти са много по-богати от тези на Visual Basic и напълно достатъчни за начално обучение, особено на ученици, запознати с Паскал [3];
- палитрата с визуални компоненти на Delphi е много разнообразна и тяхното използване е изключително удобно [2,6];
- простотата на езика е съчетана с бързина на компилирането и ефективност на компилираната програмна единица.

В модела се предлагат две нива на усвояване на ООП – въвеждащо и приложно. В първо ниво се усвояват основните понятия в ООП и се правят „конзолни“ програми. Във второто ниво се използват възможностите на визуалното програмиране, комбинирано с елементи на компютърната графика. Реализират се обектно-ориентирани програми за изобразяване на равнинни криви и компютърна анимация.

1. Обектно-ориентирано програмиране. Възникнало в края на шестдесетте години на двадесети век, утвърдило се през осемдесетте години и особено бурно развило се през последното му десетилетие, ООП се наложи като водещ стил на програмиране в началото на новия век [1].

1.1. Основни понятия в ООП. Основно понятие в ООП е понятието **клас**. Класът представлява съвкупност от обекти с еднакви физически и функционални характеристики. Функционалните характеристики (наречени още **методи**) определят „поведението“ на обектите от всеки клас. Методите позволяват да се извършват определени операции над обектите. Извикването на даден метод за конкретен обект се нарича още предаване на **съобщение** към този обект.

Класовете се организират в **йерархия**. Обикновено тя е резултат от определена класификация в някоя предметна област. Свойството, при което характеристиките на даден клас се унаследяват от друг клас, се нарича **наследяване**. То е едно от основните свойства на класовете в ООП. Класът, който унаследява характеристики от своя **предшественик**, се нарича **наследник**. Наследниците биват преки и непреки. Съвместимостта по присвояване е в посока отдолу-нагоре по йерархията. На инстанции от клас, който е по-нагоре в йерархията могат да се присвояват инстанции от клас, който е по-ниско в йерархията и е негов пряк или непряк наследник.

Друго съществено свойство е **капсулирането (херметизация)**. Според него обектите се разглеждат като херметични механизми, достъпът до които се осъществява посредством съобщения (извикване на методи). Вътрешното им устройство не е необходимо да бъде познато.

Третото важно характерно свойство на ООП е **полиморфията**. Един и същи метод може да се прилага над обекти, принадлежащи на различни класове от йерархията. При реализирането на полиморфията отпада изискването типът на обекта да бъде известен още по време на компилирането. Реализирането ѝ се базира на факта, че този тип може да стане известен чак по време на изпълнение на програмата.

Абстрактният клас се създава с единствена цел да подготви методи, които да се ползуват в цялата йерархия. Конкретни обекти от такъв клас не се предвиждат. Той подсигурява наследствени свойства и реализира алгоритми (посредством полиморфни методи) на абстрактно ниво, общи за всички наследници.

1.2. Задачи. Примерните задачите се подбират така, че да могат да се решават на части след усвояването на всяко основно понятие. Така например след темата „Деклариране на класове от обекти“ могат да се декларират първите класове (които нямат предшественици) и да се дефинират техните методи. След „Наследяване“ – да се декларират техните наследници и т.н. След усвояване на последното основно понятие „Полиморфизъм“, тези задачи могат да се решават изцяло. Представяме две примерни задачи, които може да се използват като шаблон за други подобни [4, 5].

Задача 1. Даден е модул, в който е дефиниран клас, описващ автомобил с полета за: марка, цена за 1 км. пробег и пробег (в км.) и с методи за: инициализация стойностите на полетата, прочитане стойностите на полетата, пресмятане на:

$$\textbf{цена на курса} = \textbf{(цена за 1 км.)} * \textbf{(пробег)}$$

и пресмятане на коефициент на полезно действие (КПД) на автомобила по формулата **КПД:=(цена за 1 км.)∗ (пробег)/цена на курса**.

В програма, използваща този модул:

- Дефинирайте клас (наследник на горния), описващ таксиметров автомобил с полета за: марка, цена за 1 км. пробег, пробег (в км.), цена за 1 мин. престой и престой (в мин.) и с методи за: инициализация стойностите на полетата, прочитане стойностите на полетата, пресмятане на **цена на курса = (цена за 1 км.)∗ (пробег) + (цена за 1 мин.) ∗ (престой)** и пресмятане на КПД по същата формула.
- Дефинирайте клас (наследник на горния), описващ товарен таксиметров автомобил с полета за: марка, цена за 1 км. пробег, пробег (в км.), цена за 1 мин. престой и престой (в мин.) и с методи за: инициализация стойностите на полетата, прочитане стойностите на полетата, пресмятане на:

цена на курса=2∗(цена за 1 км.)∗(пробег)+(цена за 1 мин.)∗(престой)
и пресмятане на КПД по същата формула.

- Декларирайте и инициализирайте 3 обекта както следва:
  - автомобил Mazda с цена за 1 км. пробег 20 ст. и пробег 120 км.
  - такси Toyota с цена за 1 км. пробег 35 ст., пробег 120 км., цена за 1 мин. престой 15 ст. и престой 12 мин.
  - товарно такси Iveco с цена за 1 км. пробег 1 лв., пробег 120 км., цена за 1 мин. престой 10 ст. и престой 32 мин.
- Изведете КПД за трите автомобила.
- Направете полиморфна подпрограма, която сравнява КПД на два автомобила от произволен клас и извежда съответния текст.

Усвояването на абстрактните класове е последната и най-трудна стъпка от въвеждащото ниво.

Задача 2. В отделен модул дефинирайте абстрактен клас, описващ коледна фигурка, предназначена за облепяне със станиол. Създайте програма с йерархия от следните фигурки: триъгълник, триъгълна призма, квадрат, куб, правоъгълник, паралелепипед, кръг, сфера.

- За всеки клас от йерархията дефинирайте необходимите полета с характеристики на фигурите (напр. радиус, страна и т.н.) за изчисляване на пълната повърхнина (в кв.см.) на съответната фигурка.
- Добавете методи за изчисляване на пълната повърхнина (ПП) на всяка фигурка. Какви трябва да бъдат тези методи – статични или динамични?
- Реализирайте метод за пресмятане количеството станиол (КС) в кв.см. необходимо за една фигура по формулата **КС = ПП / (Коефициент брак)** ($0 <$ Коефициент брак < 1), където Коефициент брак се подава като формален параметър на метода;
- Направете полиморфна подпрограма, която изчислява продажна цена на определен брой обекти от един клас. Формални параметри на тази подпрограма са обект, брой, цена на 1 кв.см. станиол, печалба (в %) и търговска отстъпка (в %).
- Дефинирайте подпрограма **Menu**, която да има възможности за избор на една от фигурките, въвеждане на характеристиките ѝ и извеждане на продажна цена за въведен от потребителя брой фигурки от този вид, цена на 1 кв.см. станиол, печалба (в %) и търговска отстъпка (в %).

2. Компютърна графика. Компютърната графика се занимава с изграждането (синтеза), представянето, обработката и изобразяването на графична информация. В модела се спираме по-подробно на изобразяването на такава информация на монитор, състоящ се от множество точки (пиксели), които може да имат различен цвят.

2.1. Графики на равнинни криви. Аналитично кривите се представят по различни начини (в декартови или полярни координати, параметрично и т.н.). В модела се дефинира клас Grafika, съдържащ три полиморфни метода. Първият чертае крива, представена като функция на една променлива, вторият такава, представена параметрично и третият – в полярни координати. Функциите, чиито графики се чертаят, са абстрактни методи и се предефинират в конкретните наследници. Три

частни метода на визуалната форма чертаят съответния тип графика. Формалният параметър G на всеки метод е от тип Grafika. Като фактически параметър може да се използва обект от клас – произволен наследник на Grafika. Например полиморфният обект G в първия случай може да е инстанция от тип Sinus, във втория – от SinusH, а в третия – от NewtonTri. Трите примерни класа Sinus, SinusH и NewtonTri са наследници на Grafika, в които са дефинирани съответно синус, хиперболичен синус и тризъбец на Нютон.

2.2. Компютърна анимация. В тази част се демонстрират някои елементарни техники за компютърна анимация, която се използува изключително при създаването на различни игри, каквито повечето ученици познават.

Най-известната техника за „раздвижване“ на даден обект, е посредством начертаването му, скриването му и начертаването му на ново място. Базовият клас, от който произхождат всички анимирани обекти, се дефинира в отделен модул. Тези обекти се рисуват спрямо относителна координатна система (ОКС) върху платно **C** с център **x0**, **y0**. Конструкторът инициализира тези полета, методът **DrawFunc** рисува, а **EraseFunc** скрива обекта. И двата метода са виртуални, защото са специфични за всеки рисуван обект. В модела се използват веднъж за окръжност, след това за правоъгълник, парашут, картинка (BMP) и обекти от тип **TShape**, които динамично се създават и унищожават по време на изпълнение на програмата а не при дизайна ѝ. Методът, който премества обекта на ново място с нови координати на центъра на ОКС **newx** и **newy**, е **MoveObj**. Той моделира преместването като първоначално скрива обекта посредством метода EraseFunc. След това присвоява нови координати на центъра на ОКС (x0 := newx; y0 := newy) и рисува обекта на новото място, използвайки метода DrawFunc.

И така съвсем икономично се декларира абстрактен клас, който е базов за всички анимирани обекти. Остава да се дефинират конкретни наследници и да се движат посредством един полиморфен обект.

Основната идея при вертикалното „пускане“ на обектите е да се инициализира падащият обект след което за падане надолу в цикъл се увеличава ординатата **y** с някаква стъпка **step**, а абсцисата остава постоянна. Предвидено е и падане с отместване. Тогава абсцисата x се мени по случаен начин като към нея се прибавя число от -2 до 2 ($\mathbf{x} := \mathbf{x} + \mathbf{random(5)} - \mathbf{2}$).

Разновидност на първата техника се реализира посредством използването на фон (различен от бял), на който се извършва движението. Наличието на фон позволява скриването на обекта да става посредством рисуване на този фон върху платното. В този случай методът **EraseFunc** се свежда единствено до зареждане на фона върху платното.

Докато в първите два случая рисуването на кръг и правоъгълник е тривиална задача, защото има вградени методи за тази цел, то значително повече усилия изисква рисуването на парашута (методът **DrawFunc**).

Четвъртият обект използва картинки във формата BMP. Той е наследник на парашута и в две частни полета пази необходимите картинки.

Последният обект е фигура – визуален компонент на средата (с частно поле **private Sh:TShape**). Създаването на този контрол става динамично (по време на изпълнение на програмата). Изрисуването на фигурата изисква изчисляване на стойностите на характеристиките **Left** и **Top**, които определят местоположението на

фигурата спрямо фона. След което методът **Show** я визуализира. За скриването ѝ е достатъчно само да се извика методът **Hide**.

За да се анимира друга фигура от тип **TShape** е достатъчно да се дефинира наследник с единствен метод конструктор, в който се сменя само видът на фигурата.

В последния метод за анимация, освен фона, анимираният обект използва още едно поле, което може да се оприличи на сцена с пусната завеса. Първоначално фонът се копира върху сцената. След това се копират по познатия начин анимираните обекти върху сцената. Накрая „се вдига завесата“ – сцената се копира върху платното.

ЛИТЕРАТУРА

[1] Г. Буч. Объектно-ориентированное проектирование с примерами применения. Москва, Конкорд, 1992

[2] М. Канту. Mastering Delphi 6. С., Софтпрес, 2002.

[3] Хр. Крушков, А. Илиев. Практическо ръководство по програмиране на Паскал. Пловдив, Макрос, 2003.

[4] Хр. Крушков. Програмиране с Delphi. Част I – Въведение в обектно-ориентираното програмиране. Пловдив, Макрос, 2004.

[5] Хр. Крушков, П. Иванова, К. Иванов. Практическо ръководство по програмиране на Паскал. Част III – Компютърна графика. Обектно-ориентирано програмиране. Пловдив, Анима, 2000.

[6] С. Тейхера, Х. Пачеко. Delphi 6. С., Инфодар, 2000

ПУ „Паисий Хилендарски“
ФМИ, катедра „Компютърна Информатика“
ул. „Цар Асен“ 24, 4000 гр. Пловдив
e-mail: hdk@pu.acad.bg, mik@pu.acad.bg

AN INTRODUCTION TO OBJECT-ORIENTED PROGRAMMING WITH OBJECT PASCAL AND DELPHI

Hristo D. Krushkov, Mariana Iv. Krushkova

Three programming languages are typically used in Bulgarian high school education – Basic, Pascal and C. The main courses make no provision for object-oriented programming learning. This lack of skills in the education of the gifted students could be compensated by elective courses. In the presented paper a model for introducing the object-oriented programming with Object Pascal and the environment for visual programming Delphi for these courses is described.