Educational Technologies
Образователни технологии

ОТ СТРУКТУРНО КЪМ ОБЕКТООРИЕНТИРАНО ПРОГРАМИРАНЕ

Христо Крушков
Пловдивски университет „Паисий Хилендарски"

Резюме. Възникнало в края на шестдесетте години на XX век, утвърдило се през осемдесетте години и особено бурно развило се през последното му десетилетие, обектоориентираното програмиране (ООП) се наложи като водещ стил на програмиране в началото на новия век. И докато този стил на програмиране е почти безалтернативен в масовото производство на софтуер, внедряването му в средношколската образователна практика закъснява. В статията са анализирани причините за бавното навлизане на обектоориентираното програмиране в средното училище и е представен подход за смяна на стила на преподаване на дисциплината „Програмиране". Подходът е илюстриран с примерно решение на задача от кандидатстудентски изпит по информатика в ПУ „Паисий Хилендарски", използвайки C#.

Keywords: computer science education, object-oriented programming, teaching programming, teaching methodology

Въведение

Обектоориентираното проектиране и програмиране поставиха началото на нова ера в създаването на софтуер. Успешното им съчетаване с трите основни парадигми – процедурна, логическа и функционална, както и със СУБД, е още едно свидетелство за това. Внедряването им в средношколската образователна практика обаче закъснява. Анализ на решения на задачи от кандидатстудентски изпит по информатика през последните три години показва, че над 80% от кандидатите използват т. нар. класически подход, в основата на който стои процедурното структурно програмиране. Причините за бавното навлизане на обектоориентираното програмиране в средното училище са както обективни, така и субективни. Към обективните, на първо място, стои огромната инертност на образователната система. Разработените през 2000 и актуализирани през 2006 г. държавни образователни изисквания за учебно съдържание по „Информатика" и „Информационни технологии" позволяват в обучението „да се прилага както класически, така и обектоориентиран подход за създаване на програмни продукти". Като прибавим и малкия брой

часове по информатика в учебния план за задължителна подготовка, се вижда, че основната тежест за смяна на стила на преподаване на програмиране – от структурен към обектоориентиран, пада върху учителя. Това означава, че той трябва, на първо място, сам да овладее този стил на програмиране, да подготви нови учебни материали и методика за преподаването им. Мотивацията да се свърши този голям обем творческа работа, е оставена в ръцете на училищните ръководства и ентусиазма на учителите. Като отчетем нарастващата средна възраст на учителите и намаляващия брой квалифицирани млади преподаватели, мотивирани да преподават информатика, проблемът наистина е тежък.

Университетските преподаватели не можем да се похвалим с подготовката на учебници и помагала, подходящи за различни нива на подготовка по програмиране на база C# и Java. Единственият учебник по информатика за IX – X клас, в който се поставят основите на обектоориентираното програмиране със C#, е издаден през 2013 г. (Manev et al., 2013). При обучение предимно се използват книгите на Светлин Наков и колектив (Nakov et al., 2008; Nakov et al., 2009), които се разпространяват и безплатно в електронен вариант.

Изискването на кандидатстудентски изпит по информатика в ПУ бе програмата да се пише на един от езиците Pascal, C, C++, Basic. Едва от 2014 г. бяха добавени C# и Java. Изоставането по отношение на разширяване възможностите за прилагане на ООП посредством новите ЕП е очевидно. Авторът на тази статия също търпи критика за забавянето и се надява с нея да стартира наваксването.

Смяна на стила на преподаване

Структурното програмиране заменя машинноориентирания стил на създаване на програми, в който се използва активно операторът за безусловен преход "goto". То създава удобства за проектиране „от горе надолу", модулност посредством използване на подпрограми и подходящи управляващи конструкции. Този качествен скок се усеща от самите разработчици на софтуер, улеснени в проектирането, създаването, тестването и актуализирането на програмни продукти. Структурният стил на програмиране е доминиращ почти до края на миналия век.

Както винаги, образованието изостава от съвременните тенденции и с масовото навлизане на компютър „Правец" в България през осемдесетте години голяма част от обучаемите усвоиха умения за работа с Бейсик и писане на „спагети код". Впоследствие през деветдесетте се разви успешно навлизането на структурното програмиране в училище предимно с използването на Turbo Pascal. Голяма част от учителите обаче продължиха да работят с Бейсик, но вече с по-нови версии на „Майкрософт". В

повечето случаи не се използваха главните предимства на структурното програмиране, като основен недостатък беше липсата на модулност. Това ясно проличаваше при решаване на задачите от конкурсния изпит по програмиране в ПУ „Паисий Хилендарски". Кандидат-студентите обикновено събираха целия програмен текст в рамките на една главна програма, въпреки че подусловията на задачата явно обособяваха отделните програмни части. Научени на такъв стил на програмиране, вече станали студенти, те трудно успяваха в първи курс да работят с подпрограми. Изключение правеха подготвяните в кандидатстудентските курсове по информатика бъдещи студенти, които усвояваха основите на структурното програмиране по време на курса.

Наскоро представихме[1)] примерно решение на конкурсна задача по информатика, решена в стил структурно програмиране със C/C++. В настоящата статия ще решим същата задача в обектоориентиран стил със C#. Ще покажем, че е възможен плавен преход между двата стила, като основната разлика е в проектирането на основните компоненти на програмата, докато използваните управляващи конструкции са максимално близки и запазват императивния си характер.

Реалните предимства на ООП се проявяват при по-големите софтуерни проекти. При по-малки задания, и особено при решаването на задача от конкурсен изпит, тези предимства не могат да бъдат усетени. В училище също няма достатъчно време за големи проекти. Това е и главната причина за бавното навлизане на ООП в учебната практика. Набляга се на стандартни типове данни, операции с тях, библиотечни функции, управляващи конструкции и структури от данни. На това ниво няма особено значение с какъв език и в какъв стил ще бъдат практикувани. Започването на обучението по програмиране в обектоориентиран стил обаче би донесло една дисциплина на проектиране, която, добре усвоена в началото, бавно и полека ще носи дивиденти на обучаемите в изкачването на стълбата на професионалното програмиране. Голяма част от проблемите при смяна на стила на програмиране и преподаване са детайлно описани в (Hristov, 2010; Hristov, 2011). Тук ще развием един практически подход, илюстриран с три варианта на решение. Да си припомним условието на задачата.

Условие: Да се състави компютърна програма, подпомагаща обработване на данни за резултатите от изпит по информатика в група от до 40 студенти. За целта:

1. За всеки студент да се въведе следната информация: факултетен номер (знаков низ до 10 знака), име (знаков низ до 40 знака, съдържащ трите имена на студента, разделени с точно един интервал), оценка от изпита (цяло число), име на изпитващ преподавател (знаков низ до 40

знака съдържащ трите имена на преподавателя разделени с точно един интервал);

2. Да се изведе списък на всички студенти, съдържащ факултетен номер, име на студент, оценка от изпита и име на изпитващ преподавател. Списъкът да бъде подреден по факултетен номер (в нарастващ ред). Полетата да бъдат разделени със запетая и един интервал, а името на преподавателя да излиза във вида инициали на името и презимето и фамилия. Например:

0601261050, Иван Димов Колев, 5, Е. П. Гоцев

3. Да се изведе списък на всички студенти, съдържащи в името си „Димо" или „димо". Списъкът да бъде подреден по оценка в низходящ ред, а тези с една и съща оценка – по име на студент (в азбучен ред);

4. Да се въведе информация за три групи от студенти, като се контролира броят на студентите във всяка група да не надхвърля 40. За всяка от тях:

а) да се изведат справките от точки 2 и 3;

б) да се пресметне и изведе на екрана средният успех на студентите, изпитвани от преподавател Е. П. Гоцев.

Да се намери и отпечата най-ниският от трите средни успеха.

Тази задача дава добра представа за нивото, което трябва да се покрие от кандидат-студентите. Те трябва да покажат знания и умения за работа със структури от данни и алгоритми, залегнали в учебните програми. Задачата позволява да се използва както „класически", така и обектоориентиран подход. Дава възможност за демонстрация на естествен преход от единия към другия. Най-добре в началото да се представи решение, което не използва всички предимства на ООП (като капсулиране напр.), а да се наблегне на проектирането и реализирането на класове и обекти. Така обучаемите ще успеят да осмислят превръщането на подпрограмите в локални за класа методи. Ще забележат, че няма съществена разлика при реализацията им, а само в параметрите. Обикновено методите на класовете използват директно полетата на класа, а за подпрограмите тези данни са параметри. Важно е също първоначално да разбере и ролята на метода конструктор. Впоследствие може да се демонстрират капсулирането, удобните методи за вградена сортировка, че даже и възможностите за реализиране на заявки.

Примерни решения на C#

В решенията дефинираме два класа – студент и група от студенти. Полетата на класа Student следват полетата на структурата от класическото

решение. Функцията за въвеждане на данни за един студент е заменена от метод конструктор без параметри: Student(). Функцията за извеждане на данни за един студент се заменя от метода OutStudent(). Разликата в дефинициите на методите и функциите е обусловена единствено от разликата в синтаксиса за вход/изход при различните езици за програмиране. Класът Group има едно поле – масив от студенти (групата), конструктор Group(), в който се въвежда действителният брой студенти с контрол на стойността на този брой, методи за въвеждане и извеждане на данни за групата студенти. Тези методи алгорит-мично са копия на функциите, реализирани на C/C++, само че нямат параметри. Масивът, който се обработва, е поле на класа, а действителният брой студенти се извлича динамично.

Ще представим три варианта на решение на задачата, тествани с SharpDevelop, версия 3.2.1.6466. В първия вариант се използват публични полета, а главният метод последователно обработва трите групи. Този вариант е най-подходящ за начинаещи. Тези, които имат представа от структурно програмиране, ще го възприемат по-лесно.

```
using System;
namespace Izpit2014A // Вариант 1
{
      class Student
      {
        public string fnom, // факултетен номер
                      ime, // име на студент
                      prep,// име на преподавател
                   iniprep;// име на преподавател с инициали
        public int ocenka;  // оценка
        public Student()
         {// Въвеждане на данни за един студент
             Console.Write("Въведете ф.номер:"); fnom=Console.ReadLine();
             Console.Write("Въведете име на студент:"); ime=Console.ReadLine();
             Console.Write("Въведете оценка:");
             ocenka=int.Parse(Console.ReadLine());
             Console.Write("Въведете име на преподавател:");
             prep=Console.ReadLine();
             int s1=prep.IndexOf(" "), // Индекс на първи интервал
             s2=prep.LastIndexOf(" "); // Индекс на последен интервал
             if (s1<0||s2<0) return;
```

```
        s1++; // s1 е индекс на първа буква на второ име
        s2++; // s2 е индекс на първа буква на фамилия
        iniprep=prep[0]+". "+prep[s1]+". "; // Инициали
        iniprep=iniprep+prep.Substring(s2); // Долепя отдясно фамилията
    } //Student
    public void OutStudent()
     {  // Извеждане на данни за един студент
          Console.WriteLine(fnom+", "+ime+", "+ocenka+", "+ iniprep);
     } //OutStudent
};
class Group
{
        Student[] a;
        public Group()
        {   int br;
          do
          {
                  Console.Write("Въведете брой студенти:");
                  br=int.Parse(Console.ReadLine());
          }  while (br<=0||br>40);
          a=new Student[br];
        }
        public void Input()
        {  // Въвеждане на масив от студенти
          for (int i=0; i<a.Length; i++)
          {
             Console.WriteLine("Въведете данни за "+(i+1)+"-ия студент");
              a[i] = new Student();
          }
        } //Input
        public void Output()
        {  // Извеждане на масив от студенти
          for (int i=0;i<a.Length;i++)
                  a[i].OutStudent();
        } //Output}
        public void SortFnom()
        {  // Възходяща сортировка по ф.номер
          for (int i=1; i<a.Length; i++)
```

```
            for (int j=a.Length-1; j>=i; j--)
                    if (String.Compare(a[j-1].fnom,a[j].fnom)>0)
                    {
                            Student pom=a[j];
                            a[j]=a[j-1];
                            a[j-1]=pom;
                    }
    } //SortFnom
    public void SortOc() // Низходяща сортировка по успех
    {   // При еднакъв успех, възходяща по име на студент
      for (int i=1; i<a.Length; i++)
            for (int j=a.Length-1; j>=i; j--)
                    if (a[j-1].ocenka<a[j].ocenka||
                       a[j-1].ocenka==a[j].ocenka&&
                       String.Compare(a[j-1].ime,a[j].ime)>0)
                    {
                            Student pom=a[j];
                            a[j]=a[j-1];
                            a[j-1]=pom;
                    }
    } //SortOc
    public void Spravka2() // Извеждане на студенти с Димо/димо
    {
      for (int i=0;i<a.Length;i++)
            if (a[i].ime.IndexOf("Димо")>=0||
               a[i].ime.IndexOf("димо")>=0)
                    a[i].OutStudent();
    } //Spravka2
    public double Spravka3() //Изчисляване на среден успех при
    {                                  //      "Е. П. Гоцев"
      int count=0;double sum=0;
      for (int i=0;i<a.Length;i++)
      {
            if (a[i].iniprep=="Е. П. Гоцев")
            { sum+=a[i].ocenka; count++;}
      }
      if (count>0) return sum/count; else return 0;
    } //Spravka3
```

```
    }
    class Program
    {
        public static void Main(string[] args)
        {
          Group a = new Group();
          a.Input();
          Console.WriteLine("Списък на I група подреден по ф.номер");
          a.SortFnom(); a.Output();
          Console.WriteLine("Студенти от I група с Димо/димо в името");
          a.SortOc();    a.Spravka2();
          Group b = new Group();
          b.Input();
          Console.WriteLine("Списък на II група подреден по ф.номер");
          b.SortFnom(); b.Output();
          Console.WriteLine("Студенти от II група с Димо/димо в името");
          b.SortOc();    b.Spravka2();
          Group c = new Group();
          c.Input();
          Console.WriteLine("Списък на III група подреден по ф.номер");
          c.SortFnom(); c.Output();
          Console.WriteLine("Студенти от III група с Димо/димо в името");
          c.SortOc();    c.Spravka2();
          double sr1=a.Spravka3();
          Console.WriteLine("Ср.успех при Е. П. Гоцев на I група:"+sr1);
          double sr2=b.Spravka3();
          Console.WriteLine("Ср.успех при Е. П. Гоцев на II група:"+sr2);
          double sr3=c.Spravka3();
          Console.WriteLine("Ср.усп. при Е. П. Гоцев на III група:"+sr3);
          double min=sr1;
          if (sr2<min) min=sr2;
          if (sr3<min) min=sr3;
          Console.WriteLine("Минимален ср.успех при Е. П. Гоцев:"+min);
          Console.Write("Натиснете клавиш . . . ");
          Console.ReadKey(true);
        }
    }
}
```

Вторият вариант използва частни полета и публични методи-екстрактори за извличане на стойностите на тези полета. Името с инициалите на преподавателя не се съхранява постоянно като поле, а се извлича от метода **IniPrep**(). Обхождането на всички елементи на масива е реализирано посредством оператора **foreach**. За сортиране се използва методът **Sort** на класа **Array**, като първи параметър е масивът, който се сортира, а втори – т.нар. делегат (обект-метод), който е подобен на указател към функция в C/C++. Делегатът сравнява два обекта по определени критерии и връща подобно на метода **String.Compare** отрицателна, нулева или положителна целочислена стойност. Ако делегатът върне положителна стойност при сравнение на двата обекта, то те са неправилно подредени и сортировката трябва да смени местата им. Използването на вградени методи за сортиране и делегати не е за препоръчване да се дава на начинаещи, но преподавателят винаги трябва да има възможност да представи повече варианти на напредналите ученици. Последната разлика в това решение е добавянето на поле **grID**-идентификатор на групата в класа **Group**. Инициализира се чрез входния параметър на конструктора Group: **public Group**(string groupID). Това позволява в класа Program да се добави метод **Run**(Group x), който реализира всички обработки на една група. В главния метод **Run** се вика три пъти за всяка група.

```
using System;
namespace Izpit2014A2 // Вариант 2
{
     class Student
     {
       string fnom, // факултетен номер
                    ime, // име на студент
                    prep;// име на преподавател
       int ocenka;  // оценка
       public string GetFnom() {return fnom;}
       public string GetIme() {return ime;}
       public string GetPrep() {return prep;}
       public int GetOcenka() {return ocenka;}
       public Student()
        {// Въвеждане на данни за един студент
              Console.Write("Въведете ф.номер:"); fnom=Console.ReadLine();
              Console.Write("Въведете име на студент:"); ime=Console.ReadLine();
              Console.Write("Въведете оценка:");
              ocenka=int.Parse(Console.ReadLine());
              Console.Write("Въведете име на преподавател:");
              prep=Console.ReadLine();
        } //InStudent
       public string IniPrep()
        { // връща име на преподавател с инициали
              int s1=prep.IndexOf(" "), // Индекс на първи интервал
```

```
            s2=prep.LastIndexOf(" "); // Индекс на последен интервал
            if (s1<0||s2<0) return " ";
            s1++; // s1 е индекс на първа буква на второ име
            s2++; // s2 е индекс на първа буква на фамилия
            string iniprep=prep[0]+". "+prep[s1]+". "; // Инициали
            iniprep=iniprep+prep.Substring(s2); // Долепя отдясно фамилията
            return iniprep;
      } //IniPrep
     public void OutStudent()
      {  // Извеждане на данни за един студент
            Console.WriteLine(fnom+", "+ime+", "+ocenka+", "+ IniPrep());
      } //OutStudent
};
class Group
{
            Student[] a;
            string grID;
            public string GetGrID(){return grID;}
            public Group(string groupID)
            {   int br;
                do
                {
                     Console.Write("Въведете брой студенти за "+groupID+" група:");
                     br=int.Parse(Console.ReadLine());
                }  while (br<=0||br>40);
                a=new Student[br];
                grID=groupID;
            } //Group
            public void Input(){...}// Същият като във вариант 1
            public void Output()
            {   // Извеждане на масив от студенти
                     foreach (Student x in a)
                              x.OutStudent();
            } //Output
            public void SortFnom()
            {   // Възходяща сортировка по ф.номер
                     Array.Sort(a, delegate(Student x, Student y)
                               { return String.Compare(x.GetFnom(),y.GetFnom());});
            } //SortFnom
            public void SortOc() // Низходяща сортировка по успех
            {   // При еднакъв успех, възходяща по име на студент
                     Array.Sort(a, delegate(Student x, Student y)
                              { if (x.GetOcenka()!=y.GetOcenka())
                                                  return y.GetOcenka()-x.GetOcenka();
                                else return String.Compare(x.GetIme(),y.GetIme());});
            } //SortOc
            public void Spravka2()
            { // Извеждане на студенти с Димо/димо
                     foreach (Student x in a)
```

```
                        if (x.GetIme().Contains("Димо") ||
                           x.GetIme().Contains("димо") )
                                x.OutStudent();
            } //Spravka2
            public double Spravka3()
            { //Изчисляване на среден успех при "Е. П. Гоцев"
                    int count=0;double sum=0;
                    foreach (Student x in a)
                    {
                            if (String.Compare(x.IniPrep(),"Е. П. Гоцев")==0)
                            { sum+=x.GetOcenka(); count++;}
                    }
                    if (count>0) return sum/count; else return 0;
            } //Spravka3
    }
    class Program
    {
            public static double Run(Group x)
            {
                    x.Input();
                    Console.WriteLine("Списък на "
                    +x.GetGrID()+" група подреден по ф.номер");
                    x.SortFnom(); x.Output();
                    Console.WriteLine("Студенти от "+x.GetGrID()
                    +" гр. с Димо/димо в името");
                    x.SortOc();       x.Spravka2();
                    double sr=x.Spravka3();
                    Console.WriteLine("Ср.успех при Е. П. Гоцев на "
                    +x.GetGrID()+" група:"+sr);
                    return sr;
            } //Run
            public static void Main(string[] args)
            {
                    Group a=new Group("I");  double min=Run(a);
                    Group b=new Group("II"); double sr=Run(b);
                    if (sr<min) min=sr;
                    Group c=new Group("III"); sr=Run(c);
                    if (sr<min) min=sr;
                    Console.WriteLine("Минимален ср.успех при Е. П. Гоцев:"+min);
                    Console.Write("Натиснете клавиш . . . ");
                    Console.ReadKey(true);
            } //Main
    }

}
```

Третото решение демонстрира използването на възможностите на интегрирания в C# език за заявки (Language-Integrated Query). Изисква деклариране на използването на System.Linq: **using** System.Linq. За улес-

нение при тестването данните могат да се четат от текстов файл, като първият ред е число, показващо броя на студентите, а всяко поле е на отделен ред. Текстовият файл се разполага в папката, където е изпълнимият файл, и може да се разглежда и редактира отделно с редактор, но трябва да се спазва съответният формат. Името на файла се формира автоматично от стринга „File" с долепен идентификатор на групата и разширение txt. Декларацията за работа с файлове е **using** System.IO. Добавен е втори конструктор, който чете полетата от текстовия файл: **public Student**(StreamReader file), както и методи за четене от файла и запис в него.

```
using System;
using System.Linq;
using System.IO;
namespace Izpit2014A3 // Вариант 3
{
     class Student
     {
       string fnom, // факултетен номер
                    ime, // име на студент
                    prep;// име на преподавател
       int ocenka;  // оценка
       public string GetFnom() {return fnom;}
       public string GetIme() {return ime;}
       public string GetPrep() {return prep;}
       public int GetOcenka() {return ocenka;}
       public Student() {...}// Същият като във вариант 2
       public Student(StreamReader file)
        {   //Чете полетата от текстов файл
               fnom=file.ReadLine();
               ime=file.ReadLine();
               prep=file.ReadLine();
               ocenka=int.Parse(file.ReadLine());
        }
       public string IniPrep() {...}// Същият като във вариант 2
       public void OutStudent(){...}// Същият като във вариант 2
       public void OutStudentInFile(StreamWriter file)
        {  // Извеждане на данни за един студент във файл
             file.WriteLine(fnom+"\r\n"+ime+"\r\n"+prep+"\r\n"+ocenka);
        } // OutStudentInFile
     };
     class Group
     {
             Student[] a;
             string grID;
             public string GetGrID(){return grID;}
             public Group(string groupID)
```

```
{   int br;
    grID=groupID;
    if (!File.Exists("File"+ grID +".txt"))
     do
     {
         Console.Write("Въведете брой студенти за "+ grID +" група:");
         br=int.Parse(Console.ReadLine());
     }  while (br<=0||br>40);
     else
     {
         StreamReader file = new StreamReader("File"+ grID +".txt");
         br=int.Parse(file.ReadLine());
         file.Close();
     }
    a=new Student[br];
} //Group
public void Input() {...}// Същият като във вариант 2
public void InputFromFile()
{   // Въвеждане на масив от студенти
         StreamReader file = new StreamReader("File"+grID+".txt");
         int br=int.Parse(file.ReadLine());
         for (int i=0; i<a.Length; i++)
         {
                 a[i] = new Student(file);
         }
         file.Close();
} //InputFromFile
public void OutputToFile()
{   //Записва данните на студентите в текстов файл
         StreamWriter file = new StreamWriter("File"+grID+".txt");
         file.WriteLine(a.Length);//Записва броя студенти в първи ред
         foreach (Student x in a)
                 x.OutStudentInFile(file);
         file.Close();
} //OutputToFile
public void Spravka1()
{   // Извеждане на всички студенти подредени по фак.номер
         var queryFnom = from x in a
                 orderby x.GetFnom() ascending
                 select x;
         foreach (Student x in queryFnom) x.OutStudent();
}//spravka1
public void Spravka2()
{   // Извеждане на студенти с Димо/димо
         // подредени по оценка низходящо
         // а при равни оценки - по име възходящо
         var queryDimo = from x in a
                 where x.GetIme().Contains("Димо") ||
                       x.GetIme().Contains("димо")
```

```
                        orderby x.GetOcenka() descending
                        orderby x.GetIme() ascending
                        select x;
                foreach (Student x in queryDimo) x.OutStudent();
        }//spravka2
        public double Spravka3()
        {       //Изчисляване на среден успех при "Е. П. Гоцев"
                var queryGocev = from x in a
                        where x.IniPrep() == "Е. П. Гоцев"
                        select x.GetOcenka();
                if (queryGocev.Count()>0)
                        return queryGocev.Average();
                else return 0;
        }//spravka3
}
class Program
{
        public static double Run(Group x)
        {
                if (File.Exists("File"+x.GetGrID()+".txt")) x.InputFromFile();
                else { x.Input(); x.OutputToFile();}
                Console.WriteLine("Списък на "+x.GetGrID()
                +" група подреден по ф.номер");
                x.Spravka1();
                Console.WriteLine("Студенти от "+x.GetGrID()
                +" гр. с Димо/димо в името");
                x.Spravka2();
                double sr=x.Spravka3();
                Console.WriteLine("Ср.успех при Е. П. Гоцев на "
                +x.GetGrID()+" група:"+sr);
                return sr;
        } //Run
        public static void Main(string[] args) {...}// Като във вариант 2
}
}
```

Заключение

Статията е предназначена предимно за учители, които трябва да сменят стила на програмиране и преподаване при обучение на учениците. Предложен е метод за активно обучение чрез практика – learning by doing (Krushkova, 2014), като е използвана задачата от първия кандидатстудентски изпит по информатика в ПУ „Паисий Хилендарски", когато е обявено официално в указанията към задачата използването на C# и Java. Задачите от този тип позволяват изписването на лист – едно от условията, на които трябва да отговарят задачи от писмен конкурсен изпит, и въпреки малкия си размер дават възможност да бъдат реализирани както с „класически", така и с обектоориентиран подход. Трите варианта на решение свидетелстват за разнообразните средства на C#. Решенията са възможно най-кратки и оп-

ростени, като целят да предложат на учителите средства за плавен преход от структурно към обектоориентирано програмиране. Статията би могла да подпомогне и ученици, които се интересуват от конкурсни задачи за кандидатстудентски изпит по информатика и имат много добри базови знания по програмиране.

На сайта на ФМИ при Пловдивския университет „Паисий Хилендарски" могат да се видят всички конкурсни задачи[2]. Решения на задачите и тестовете от конкурсните изпити по информатика са достъпни на уеб адрес http://hristokrushkov.com след регистрация. За читателите на списанието е осигурен специален достъп в сравнение с обикновен регистриран потребител посредством username: MatInf и парола: informatika. Там могат да намерят решения на конкурсни задачи както на C#, така и на други езици за програмиране.

NOTES / БЕЛЕЖКИ

1. Сп. „Математика и информатика", бр. 3/2016
2. http://fmi-plovdiv.org/index.jsp?id=324&ln=1

REFERENCES / ЛИТЕРАТУРА

Darzhavni obrazovatelni iziskvaniya za uchebno sadarzhanie po Informatika i Informatsionni tehnologii. (2000, 13.06.). Darzhaven vestnik. *[Държавни образователни изисквания за учебно съдържание по Информатика и Информационни технологии*. (2000, 13.06.). Държавен вестник.]

Krushkova, M. (2014). *Metodika za aktivno obuchenie po programirane chrez izpolzvane na informatsionni i komunikatsionni tehnologii.* Disertatsionen trud za prisazhdane na obrazovatelnata i nauchna stepen "doctor", Plovdiv. [Крушкова, М. (2014). *Методика за активно обучение по програмиране чрез използване на информационни и комуникационни технологии*. Дисертационен труд за присъждане на образователната и научна степен „доктор", Пловдив.]

Manev, Kr., Petrov, P., Hristova, V., Maneva, N., Yovcheva, B. & Petrov, P. (2013). *Informatika za zadalzhitelna podgotovka v IX – X klas.* Sofiya: Izkustva. [Манев, Кр., Петров, П., Христова, В., Манева, Н., Йовчева, Б. & Петров, П. (2013). *Информатика за задължителна подготовка в IX – X клас*. София: Изкуства.]

Nakov, Sv., Valkov, B., Kolev, V., Tsanev, V., Aleksiev, D., Bozhkov, L., ... & Konov, Ts. (2008). *Vavedenie v programiraneto s Java.* Veliko Tarnovo: Faber. [Наков, Св., Вълков, Б., Колев, В., Цанев, В., Алек-

сиев, Д., Божков, Л., ... & Конов, Ц. (2008). *Въведение в програмирането с Java*. Велико Търново: Фабер.]

Nakov, Sv., Georgiev, V., Kolev, V., Dimitrov D., Murdanliev, I., Yosifov, Y., ... &Konov, Ts. (2009). *Vavedenie v programiraneto sas C#.* Veliko Tarnovo: Faber. [Наков, Св., Георгиев, В., Колев, В., Димитров Д., Мурданлиев, И., Йосифов, Й., ... & Конов, Ц. (2009). *Въведение в програмирането със C#.* Велико Търново: Фабер.]

Hristov, Hr. (2011). Trudnosti i resheniya pri smyana na paradigmata. Prepodavane na obektoorientiran analiz, dizayn i programirane, *Dokladi na mezhdunarodna konferentsiya, Vzaimodeystvieto teoriya – praktika: Klyuchovi problemi i resheniya,* III, 303 – 310. [Христов, Хр. (2011). Трудности и решения при смяна на парадигмата. Преподаване на обектоориентиран анализ, дизайн и програмиране, *Доклади на международна конференция, Взаимодействието теория - практика: Ключови проблеми и решения*, III, 303 – 310.]

Hristov, H. (2010). Review and Outlooks of the Means for Visualization of Syntax Semantics and Source Code. Procedural and Object-oriented Paradigm – Differences, *In proceedings of the Anniversary International Conference REMIA*, Plovdiv, 443 – 451.

FROM STRUCTURED TO OBJECT-ORIENTED PROGRAMMING

Abstract. Arose in the late sixties of the twentieth century, affirmed through the eighties and especially booming in its last decade, object-oriented programming has established itself as a leading style of programming at the beginning of the new century. While this style of programming has almost no alternative in the mass production of software, its imposition in secondary school educational practice is delayed. The article analyses the reasons for the slow uptake of object-oriented programming in secondary school and presents an approach to change the style of teaching Programming. The approach is illustrated through a suggested solution of an entrance exam task in Informatics at the University of Plovdiv using C#.

✉ **Dr. Hristo Krushkov, Assoc. Prof.**
Department of Software Engineering
Faculty of Mathematics and Informatics
University of Plovdiv
236, Bulgaria Blvd.
Plovdiv, Bulgaria
E-mail: hdk@uni-plovdiv.bg