Educational Matters
Научно-методически статии

СЪЗДАВАНЕ НА ИГРИ В ЧАСОВЕТЕ ПО ИНФОРМАТИКА ЧРЕЗ ИЗПОЛЗВАНЕ НА ГЕНЕРАТОР НА СЛУЧАЙНИ ЧИСЛА

Емилия Николова, Даниела Тупарова
Югозападен университет „Неофит Рилски“ – Благоевград

Резюме. В статията се дават методически идеи за използване на генератора на случайни числа в обучението по програмиране. Представен е набор от задачи, свързани с изучаване на визуално програмиране на базата на езика C#, към всяка от които са изложени целите и основните етапи за решаването ѝ. Задачите илюстрират и възможност за съчетаване на процеса на обучение по програмиране с възможността за създаване на игри. Посочената група съдържа 10 задачи, от които 7 са авторски, а 3 се известни класически игри. Към всяка задача са посочени целта и основните етапи при решаването ѝ.

Keywords: computer science education; high school; problem solving in programming; game; event driven programming; random numbers generator

Въведение

Задачите играят огромна роля в живота на човек – задачи, които поставяме пред себе си, и задачи, които поставят пред нас другите хора и обстоятелства в живота ни. Всъщност мисловната дейност на човек, като цяло, е свързана с поставяне и решаване на задачи. Проблемът със задачите по информатика и ИТ е разглеждан от няколко изследователи в България. В (Dureva, 2003) Дурева разглежда методически проблеми при решаване на задачи в училищните курсове по информатика и информационни технологии. В (Asenova, 1989; Garov, 2010; Grozdev & Garov, 2008; Garov, 2004) са илюстрирани примери за системи от задачи, които могат да се използват при обучение по информатика. Проблеми, свързани с реализацията на етапите за решаване на задачи по информатика и информационни технологии, са разглеждани в (Asenova, 1989; Yadkova, 1991; Yadkova & Lazarova, 1989; Dureva, 2003).

Процесът на обучение по информатика „притежава потенциални възможности за личностно изграждане и развитие на ученика – формиране на абстрактно и логическо мислене, възпитание и формиране на адекватно отношение към заобикалящата действителност“[1]. В (Grozdev, Chehlarova & Terzieva, 2010) се отбелязва, че развитието на алгоритмичното мислене е една от ос-

новните цели на обучението по информатика. Според (Hristova & Atanasova, 2011) е добре при начално обучение по програмиране основните алгоритмични конструкции да се онагледяват със задачи, които използват числова информация и познати математически понятия, като по този начин вниманието на учениците ще се насочи пряко към осмисляне на преподавания нов материал.

Съгласно новата учебна програма по информатика за VIII клас[1)] обучението се извършва на базата на език за визуално програмиране – Visual Basic, C# или Java, по избор на преподавателя. Изисква се усвояването на необходимите знания и умения да става чрез активно участие на ученика в учебния процес. Изследвания, свързани с ролята на базовите задачи при изучаване на събитийно програмиране и системи от такива задачи, са предложени в (Aneva, 2010; Garov & Aneva, 2004). Терзиева е разработила практическо ръководство за създаване на ГПИ (Графичен потребителски интерфейс) на базата на езика C# (Terzieva, 2015), а в (Aneva, 2013) е представен модел за профилирано обучение по информатика и ИТ в гимназиален етап.

Задълбоченото изучаване и усвояване на основните принципи на събитийното програмиране е неразривно свързано с решаване на практически задачи. Основната задача пред учителите е да подберат набор от стройно изградена система от учебни задачи с различна степен на сложност, така че учениците да имат възможност да се запознаят с основните принципи и възможности на визуалното програмиране и да усвоят основни технологии и механизми за реализиране на програми, управлявани от събития с достъпен графичен потребителски интерфейс. Чрез настоящата разработка е предложен набор от задачи за изучаването на визуално програмиране на базата на езика C#, който може да се използва при организиране и провеждане на обучението по информатика в VIII клас.

Примерена група от задачи

1.1. Методически насоки за използване на задачите

Решаването на всяка от задачите преминава през следните етапи:

1) Създаване на ГПИ (графичен потребителски интерфейс)

2) Настройка на свойствата на елементите от ГПИ

3) Добавяне на код

Първите два етапа в определен случай могат да се реализират чрез използване на готов ГПИ и/или настройки в него. Това съкращава времето за работа и позволява да се акцентира върху понятия и алгоритми.

За някои задачи учителят може да постави под формата на домашна работа създаването на потребителския интерфейс и настройка свойствата на елементите му.

Всяка от задачите съдържа генериране на случайно число или символ, поради което се налага учениците предварително да са запознати с използване на класа Random. И макар че изучаването на този клас не е посочено в ДОИ по информатика, въвеждането му като помощен инструмент при решаване на различни задачи би разнообразило съдържанието на учебните задачи.

Тъй като в учебната програма, явно, не е предвидено въвеждането на понятието клас, на учениците се дава наготово конструкцията от команди за генериране на случайно число и пояснение на семантиката чрез примери.

Синтаксис: идентификатор

```
Random rnd = new Random();
```

`rnd.Next(a, b)` — *връща неотрицателно цяло число от тип Int32, принадлежащо на интервала [a, b).*

Семантика:

Примери:

```
Random r = new Random(); //На променливата r от тип Random прис-
вояваме //новосъздадена инстанция на класа Random.
int month = r.Next(1, 13); //генерира цяло число между 1 и 12
int hour = r.Next(0, 24); //генерира цяло число между 0 и 23
```

Чрез всяка от посочените задачи се развиват уменията за проектиране и изграждане на подходящ графичен потребителски интерфейс визуални средства. Освен тази цел след всяка от задачите са посочени и допълнителни специфични цели, които се постигат с решаване на съответната задача. Подредбата на задачите следва реда за изучаване на учебно съдържание, включено в новите ДОИ по информатика, и е с нарастваща сложност.

2.1. Задачи

Задача 1. Изпитване

За да избегне субективното отношение при избор на това кой да бъде изпитан, г-жа Петрова решила да използва компютърна програма, генерираща случаен номер от класа. За съжаление, тя не разполага с такава програма и моли Вас да ѝ помогнете, като създадете проект Examp1, чрез който по въведен начален и краен номер на учениците в класа се извежда кой номер ще бъде изпитан.

Задачата се използва при начално запознаване със средата за програмиране. На този етап не се дават пояснения за кода.

В темата целочислен тип с решаване на тази задача се постигат следните основни цели.

– Прилагане на усвоените знания, умения и навици за:

– проектиране на несложна форма, съдържаща етикети, текстови полета и бутони;

– настройване в режим на дизайн на основните свойства на използваните контроли;

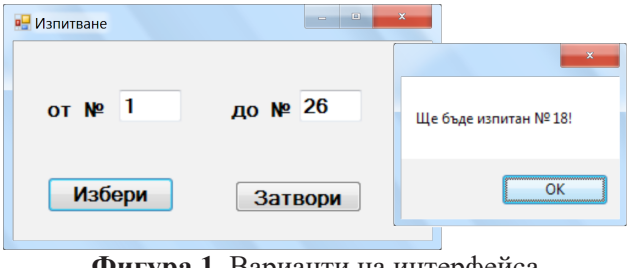
– именуване на обекти контроли и променливи съгласно общоприета конвенция;

– задаване функционалност на бутон;
– деклариране, описване и инициализация на променлива от целочислен тип;
– използване на вградени функции за преобразуване на цяло число в низ и обратно;
– използване на кутия за съобщения.

– Развиване на логическото мислене.

На учениците може да се предложи следното решение.

Първи етап – създаване на ГПИ (фигура 1).



Фигура 1. Варианти на интерфейса

Втори етап – настройка на някои свойства на елементите на ГПИ.

Таблица 1

Елемент на ГПИ	Име	Свойства
Button	btnNew	Text = "Избери"
Button	btnClose	Text = "Затвори"
Label	lblFrom	Text= "от №"
Label	lblTo	Text= "до №"
TextBox	txtFrom	Text= "1"
TextBox	txtTo	Text= "26"

Трети етап – добавяне на програмен код към елементите.

```
private void btnNew_Click(object sender, EventArgs e)
{
            Random rnd = new Random();
            int from = int.Parse(txtFrom.Text);
            int to = int.Parse(txtTo.Text);
            int number = rnd.Next(from, to + 1);
            MessageBox.Show("Ще бъде изпитан № " + number.ToString()+ "!");
}
privatevoid btnClose_Click(object sender, EventArgs e)
{
            Close();
}
```

Задача 2. Банков код

За отличните си постижения на олимпиадата по БЕЛ (отличен 5,98) Петър спечелил парична стипендия от общината. Сумата се превежда всеки месец по новата му банкова сметка. Виждайки PIN на банковата си карта, Пепи разбрал, че кодът е много лесен, и решил да го смени, като спазва следната схема:

– кодът трябва да съдържа четири цифри и първата цифра на кода да не е 0;

– втората цифра се избира произволно;

– третата цифра се получава, като се съберат първите две генерирани цифри. Ако полученият сбор е двуцифрено число, за трета цифра се взема цифрата на единиците;

– четвъртата цифра се получава, като средноаритметичното на втората и третата цифра е закръглено до цяло число.

Освен това Пепи иска всеки месец да сменя и PIN кода на телефона си. И тъй като е твърде зает с подготовката си за новата олимпиада, моли Вас за помощ, като напишете програма **PinCode**, която да генерира PIN код, отговарящ на посочените условия.

Една от особеностите на задачата е, че тя се разглежда преди въвеждането на условен оператор и решението ѝ се реализира без неговото използване. В задачата се прави връзка със знанията по математика.

Основни цели:

– Прилагане на усвоените знания, умения и навици за:

– проектиране на несложна форма, съдържаща етикети и бутони;

– настройване в режим на дизайн на основните свойства на използваните контроли;

– именуване на обекти контроли и променливи съгласно общоприета конвенция;

– задаване функционалност на бутон;

– използване на вградени функции за преобразуване на цяло число в низ.

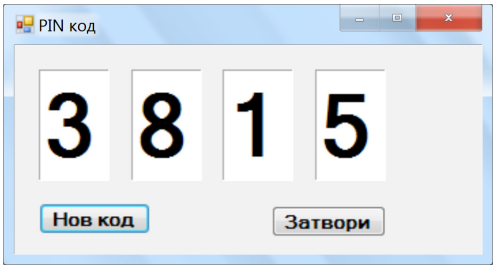
– Изграждане на умения за работа с променливи от целочислен тип данни за прилагане и анализиране на резултатите от целочислените аритметични операции и техния приоритет.

– Развиване на логическото мислене.

– Осъществяване на междупредметна връзка с математиката.

На учениците може да се предложи следното решение.

Първи етап – създаване на ГПИ на приложението (фигура 2).



Фигура 2. Варианти на интерфейса

Втори етап – настройка на някои свойства на елементите на ГПИ (таблица 2).

Таблица 2

Елемент на ГПИ	Име	Свойства
Button	btnNew	Text = "Нов код"
Button	btnClose	Text = "Затвори"
Label	lblFirst	Text="" AutoSize=False
Label	lblSecond	Text="" AutoSize=False
Label	lblThird	Text="" AutoSize=False
Label	lblFourth	Text="" AutoSize=False

Трети етап – добавяне на програмен код към елементите.

```
private void btnNewCode_Click(object sender, EventArgs e)
{
   Random rnd = newRandom();

   int first = rnd.Next(1, 10);
   int second = rnd.Next(1, 10);
   int third =  (first +second) % 10;
   int fourth = (second+third+ 1) / 2;

   lblFirst.Text = first.ToString();
   lblSecond.Text = second.ToString();
   lblThird.Text = third.ToString();
   lblFourth.Text = fourth.ToString();
}
```

Задача 3. Познай числото – класическа игра

Създайте компютърна програма, в която компютърът генерира случайно естествено число от интервала [1,100]. Потребителят трябва да познае „намисленото“ от компютъра число, получавайки от програмата отговори: „по-малко от намисленото“ или „ по-голямо от намисленото“.

Вариант на условието на задачата може да се намери в (Manev, Maneva & Hristova, 2017).

Като проект в края на учебната година или във втори модул от обучението по програмиране могат да се добавят и допълнителни изисквания към програмата:

– да извежда помощна информация за правилата на играта;

– да брои за колко хода е познато числото;

– да дава възможност на потребителя да определя интервала, от който да бъде избрано числото.

Забелязаните грешки при тестване, които биха се появили при неправилен вход, могат да се използват като предпоставка за въвеждане на оператора за обработка на изключения (Модул 2 от учебната програма).

Основни цели:

– Придобиване на умения за проектиране и изграждане на подходящ графичен потребителски интерфейс (ГПИ) и работа с етикет, команден бутон, текстово поле.

– Затвърждаване на знанията на учениците за промяна на основните свойства на използваните контроли и за задаване на функционалност на бутон.

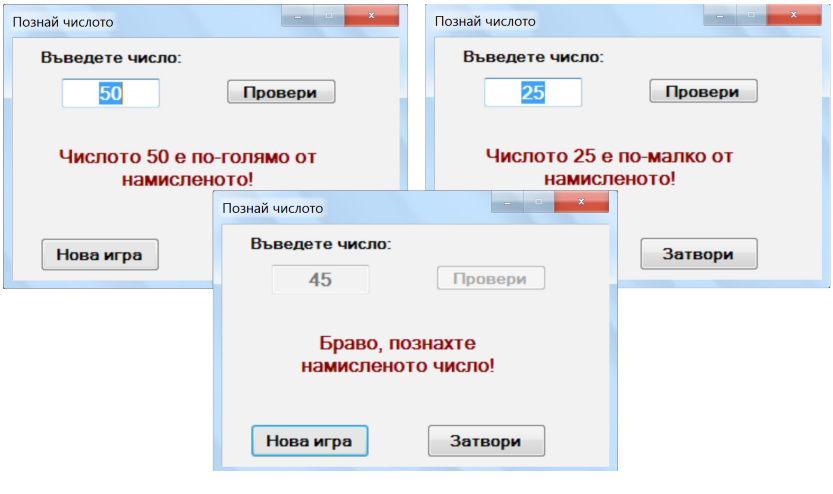
– Развиване на умения за работа с променливи, вложени условни конструкции и вградени функции за преобразуване на цяло число в низ и обратно.

– Развиване на умения за деклариране и използване на глобални променливи.

– Насочване на учениците към идеята на алгоритъма за двоично търсене в масив.

На учениците се предлага следното решение.

Първи етап – създаване на ГПИ на приложението (фигура 3).



Фигура 3. Варианти на интерфейса

Втори етап – настройка на някои свойства на елементите на ГПИ (таблица 3).

Таблица 3

Елемент на ГПИ	Име	Свойства
Button	btnNewgame	Text = "Нова игра"
Button	btnClose	Text = "Затвори"
Button	btnCheck	Text = "Провери"
TextBox	txtInput	Text=" ", TextAlign=Center
Label	lblMessage	Text="", AutoSize=False, BackColor =Maroon
label	lblType	Text= "Въведете число"

Трети етап – добавяне на програмен код към елементите.

```
int numComputer; //числото, което се "намисля" от компютъра
Random rnd = new Random();
private void Find1_Load(object sender, EventArgs e)
{
     numComputer = rnd.Next(1, 101); // генериране на число от 1 до 100
     txtInput.Text = "";
     txtInput.Focus();
}
private void btnNewGame_Click(object sender, EventArgs e)
{
     numComputer = rnd.Next(1, 101); //генериране на ново число
     lblMessage.Text = "";
     txtInput.Enabled = true;
     txtInput.Text = "";  txtInput.Focus();
     btnCheck.Enabled = true;
}
private void btnCheck_Click(object sender, EventArgs e)
{
     if (txtInput.Text != "")
     {
        int numPlayer;//числото на потребителя
        numPlayer = int.Parse(txtInput.Text);

        if (numPlayer < 1 || numPlayer > 100)
        {
           MessageBox.Show("Въведеното число е извън диапазона." +
                           " Моля, въведете число между 1 и 100!");
     }
     else
     {
          if (numPlayer == numComputer)
          {
             lblMessage.Text = "Браво, познахте намисленото число!\n";
             btnCheck.Enabled = false; txtInput.Enabled = false;
          }
          else if (numPlayer > numComputer)
          {
```

```
                lblMessage.Text = "Числото " + numPlayer.ToString() +
                                  " е по-голямо от намисленото!";
            }
            else
            {
                lblMessage.Text = "Числото " + numPlayer.ToString() +
                " е по-малко от намисленото!";
            }
        }
}
      else          {   MessageBox.Show("Моля, въведете число!");  }
      txtInput.SelectAll();
      txtInput.Focus();
}
private void btnClose_Click(object sender, EventArgs e) {    Close();   }
```

Задача 4. Камък, ножица, хартия – класическа игра

Една от любимите игри на Иван е „Камък, ножица, хартия“. До такава степен той харесва играта, че почти всички решения взема, след като разбере дали печели в играта*. Всъщност невинаги Иван е заобиколен от приятели и затова иска да му помогнете, като напишете програма, срещу която да играе. Програмата трябва да приема неговото предположение и след това да избира на случаен принцип камък, ножица или хартия. Този, който спечели 10 кръга, печели играта.

*Камък побеждава ножица, ножица побеждава хартия, хартия побеждава камък.

Друг вариант на условието на задачата може да бъде намерен в (Momcheva, Glushkova &Marinova, 2017).

За реализиране на тази задача в рамките на 1 учебен час е целесъобразно преподавателят да подготви предварително ГПИ на приложението, а в часа учениците само да добавят код към съответните контроли.

Основни цели:

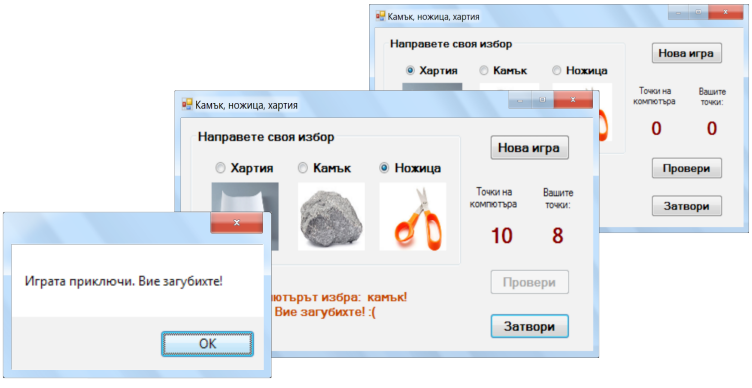
– Развитие на уменията за проектиране и изграждане на подходящ графичен потребителски интерфейс (ГПИ) и работа с етикет, команден бутон и радиобутон.

– Затвърждаване на знанията на учениците за промяна на основните свойства на използваните контроли и за задаване на функционалност на бутон.

– Обобщаване на знанията за работа с променливи, вложени условни конструкции, вградени функции за преобразуване на цяло число в низ и обратно, кутия за съобщения – Message Box.

На учениците се предлага следното решение.

Първи етап – създаване на ГПИ на приложението (фигура 4).



Фигура 4. Варианти на интерфейса

Втори етап – настройка на някои свойства на елементите на ГПИ (таблица 4).

Таблица 4

Контрол	Име на обект	Свойства
Button	btnNewGame	Text = "Нова игра"
Button	btnClose	Text = "Затвори"
Button	btnCheck	Text = "Провери"
GroupBox	groupBox1	Text = "Направете своя избор"
Label	lblCmp	Text= "Точки на компютъра"
Label	lblGuest	Text= "Вашите точки"
Label	lblResult	Text=""
Label	lblPointComputer	Text= "0"
Label	lblPointGuest	Text= "0"
RadioButton	rbPaper	Text= "Хартия"
RadioButton	rbStone	Text= "Камък"
RadioButton	rbCutter	Text= "Ножица"
PictureBox	picBoxPaper	Image= "paper.jpg"
PictureBox	picBoxStone	Image= "stone.jpg"
PictureBox	picBoxGutter	Image= "cutter.jpg"

Трети етап – добавяне на програмен код към елементите.

```
byte guest = 1, //избор на потребителя
     computer, //избор на компютъра
     pointGuest = 0,  //точки на потребителя
     pointComputer = 0;  //точки на компютъра
Random rnd = new Random();

private void btnCheck_Click(object sender, EventArgs e)
{
    computer = (byte)rnd.Next(1, 4);
    lblResult.Text = "Компютърът избра: ";

    if (computer == 1)
    {
       lblResult.Text += " хартия!\n";
       if (guest == 1)lblResult.Text += " Няма победител!  ;)";
       else if (guest == 3)
            { lblResult.Text += " Вие спечелихте!  :)";     pointGuest++;  }
            else
              { lblResult.Text += "Вие загубихте! :(";     pointComputer++;   }
}
/* аналогична проверка и за следващите два случая
*                    else if (computer == 2)  {      }
*                    else       {….   }
*/
lblPointGuest.Text = pointGuest.ToString();
lblPointComputer.Text = pointComputer.ToString();

if (pointGuest == 10)
{
   MessageBox.Show("Играта приключи. Вие спечелихте!");
   btnCheck.Enabled = false;
}
else if (pointComputer == 10)
     { MessageBox.Show("Играта приключи. Вие загубихте!");
       btnCheck.Enabled = false;
     }
}
private void btnNew_Click(object sender, EventArgs e)
{
            pointGuest = 0; pointComputer = 0;
            btnCheck.Enabled = true;
            lblPointComputer.Text = "0"; lblPointGuest.Text = "0";
            lblResult.Text = "";
}
private void rbPaper_CheckedChanged(object sender, EventArgs e)  {   guest = 1;   }
private void rbStone_CheckedChanged(object sender, EventArgs e)  {   guest = 2;   }
private void rbCutter_CheckedChanged(object sender, EventArgs e)  {  guest = 3;   }
private void btnClose_Click(object sender, EventArgs e)   {   Close();   }
```

Задача 5. Първолаци

За да провери дали учениците могат да сравняват правилно числа, учителката по математика г-жа Теоремкова решила да проведе следната игра.

На всеки ученик от класа тя дава по 4 карти, върху които има написана цифра. От тях учениците трябва да конструират най-голямото и най-малкото четирицифрено число, получено чрез разместване и долепване на картите. За съжаление, се оказало, че тази игра ѝ отнема много време от часа и затова тя ви моли да ѝ помогнете, като напишете програма, която генерира 4 цифри (поне една от които не е нула) и след това по въведени отговори на ученик за най-малко и най-голямо число проверява дали той е отговорил правилно.

Решение

Основни цели:

– Развиване на уменията за проектиране и изграждане на подходящ графичен потребителски интерфейс (ГПИ) и работа с етикет, команден бутон, текстово поле и изображения.

– Затвърждаване на знанията на учениците за промяна на основните свойства на използваните контроли и за задаване на функционалност на бутон.

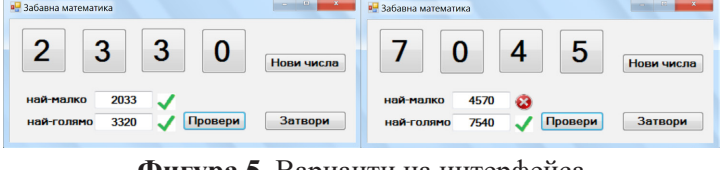
– Обобщаване на знанията за работа с променливи, вложени условни конструкции, вградени функции за преобразуване на цяло число в низ и обратно, кутия за съобщения – Message Box.

– Затвърждаване на знанията на учениците за сортиране на 4 числа.

Поставената задача може да се реализира в 2 учебни часа. Това време може да бъде съкратено, ако на учениците се предостави предварително подготвен ГПИ и направени настройки на елементите в него.

На учениците се предлага следното решение.:

Първи етап – създаване на ГПИ на приложението (фигура 5).



Фигура 5. Варианти на интерфейса

Втори етап – настройка на някои свойства на елементите на ГПИ (таблица 5).

Таблица 5

Контрол	Име на обект	Свойства
Button	btnNew	Text = "Нови числа"
Button	btnClose	Text = "Затвори"
Button	btnClose	Text = "Затвори"
Button	Button1	Text = ""
Button	Button2	Text = ""
Button	Button3	Text = ""

Button	Button4	Text = ""
Label	Lbl1	Text= "най-малко"
Label	Lbl2	Text= "най-голямо"
PictureBox	pcbMin	Image= "error.png"
PictureBox	pcbMax	Image= "preferences.png"

Трети етап – добавяне на програмен код към елементите.

За да проверим дали потребителят е въвел правилните стойности, е необходимо да сортираме „намислените“ цифри. Сортирането в низходящ ред би ни дало отговор на въпроса кое е максималното число, но за намиране на минималното трябва да съобразим, че в генерираните цифрите може да има нули. В края на събитието ***btnCheck_Click*** може да се види цялостната проверка.

```
Random rnd = new Random();
private void btnNew_Click(object sender, EventArgs e)
{
            btn1.Text = rnd.Next(1, 10).ToString();
            btn2.Text = rnd.Next(0, 10).ToString();
            btn3.Text = rnd.Next(0, 10).ToString();
            btn4.Text = rnd.Next(0, 10).ToString();
            txtMax.Text = "";
            txtMin.Text = "";
            pcbMax.Visible = false;
            pcbMin.Visible = false;
            txtMin.Focus();
}
private void btnClose_Click(object sender, EventArgs e)  { Close();  }

private void btnCheck_Click(object sender, EventArgs e)
{
             if (txtMin.Text == "")
             {
                MessageBox.Show("Въведете число!", "Грешка!");
                txtMin.SelectAll(); txtMin.Focus();
             }
             else if (txtMax.Text == "")
            {
                MessageBox.Show("Въведете число!", "Грешка!");
                txtMax.SelectAll(); txtMax.Focus();
             }
             else
             {
                int a, b, c, d;
                a = int.Parse(btn1.Text);    b = int.Parse(btn2.Text);
                c = int.Parse(btn3.Text);    d = int.Parse(btn4.Text);
                //сортиране на цифрите в нарастващ ред
                if (a > b) { int x = a; a = b; b = x; }
                if (a > c) { int x = a; a = c; c = x; }
                if (a > d) { int x = a; a = d; d = x; }
                if (b > c) { int x = b; b = c; c = x; }
                if (b > d) { int x = b; b = d; d = x; }
                if (c > d) { int x = c; c = d; d = x; }
```

```
                //намиране на най-голямото число
                string maxNumber, minNumber;
                 maxNumber = (d * 1000 + c * 100 + b * 10 + a).ToString();
                //определяне на най-малкото число
                if (a == 0 && b == 0 && c == 0) minNumber = (d * 1000).ToString();
                else if (a == 0 && b == 0) minNumber = (c * 1000 + d).ToString();
                else if (a == 0) minNumber = (b * 1000 + c * 10 + d).ToString();
                else minNumber = (a * 1000 + b * 100 + c * 10 + d).ToString();

                if (txtMin.Text != minNumber) pcbMin.Image = Properties.Resources.error;
                      else pcbMin.Image = Properties.Resources.preferences;

                if (txtMax.Text != maxNumber) pcbMax.Image = Properties.Resources.error;
                      else pcbMax.Image = Properties.Resources.preferences;

                pcbMin.Visible = true;    pcbMax.Visible = true;
            }
}
```

Задача 6. Ах, тази математика

Ето че учениците на г-жа Теоремкова са вече трети клас и могат да сравняват не само числата до 1000, а и много по-големи. И разбира се, тя отново има нужда от вашата помощ. Този път г-жа Теоремкова иска компютърът да „раздава“ на всеки ученик от класа по 4 карти, върху които има написано число от интервала [0; 999]. От тях учениците трябва да конструират най-голямото и най-малкото число, получени чрез разместване и долепване на картите, и да запишат своите предположения. И отново верният помощник – компютърът, да проверява дали ученикът се е справил с поставената задача.

Тук може да се използва вече създаденият проект в предходната задача, който да се надгради съгласно промените в условието.

Основни цели:

– Прилагане на усвоените знания, умения и навици за:

– проектиране на несложна форма, съдържаща етикети и бутони;

– настройване в режим на дизайн на основните свойства на използваните контроли;

– именуване на обекти контроли и променливи съгласно общоприета конвенция;

– задаване функционалност на бутон;

– използване на вградени функции за работа с низове;

– изграждане на умения за работа с условни конструкции.

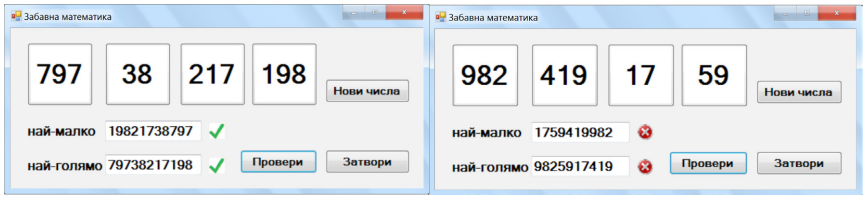
– Развиване на логическото мислене.

Решение

На учениците може да се предложи следното **решение.**

Първи и втори етап от решението съвпадат с тези на предходната задача (фигура 6).

Трети етап – добавяне на програмен код към елементите



Фигура 6. Варианти на интерфейса

За решаване на задачата трябва да съобразим, че сортиране на числата както в предходната задача няма да доведе до правилно решение. Например: 12<1200, но при долепване на картите, числото 121200 е по-голямо от 120012. Ето защо в решението е изложен алгоритъм, при който се работи с низове. За да сортираме правилно картите **a** и **b**, проверяваме дали след слепване (конкатенация) на **a** и **b** се получава по-малък низ в сравнение с този, получен след слепване на **b** и **a.**

Друг алгоритъм за решаване на тази задача може да бъде намерен в анализа на решението на задача Е1-Карти (Национален есенен турнир по информатика, Шумен, 24 – 26 ноември 2017 г.). Задачата за състезанието е предложена от Е. Николова.

```
private void btnNew_Click(object sender, EventArgs e)
{
   Random rnd = new Random();
   btn1.Text = rnd.Next(1, 1000).ToString();
   btn2.Text = rnd.Next(0, 1000).ToString();
   btn3.Text = rnd.Next(0, 1000).ToString();
   btn4.Text = rnd.Next(0, 1000).ToString();
   txtMax.Text = "";          txtMin.Text = "";
   pcbMax.Visible = false;    pcbMin.Visible = false;
   txtMin.Focus();
}

private void btnCheck_Click(object sender, EventArgs e)
{
   if (txtMin.Text == "")
   {
      MessageBox.Show("Въведете число!", "Грешка!");
      txtMin.SelectAll(); txtMin.Focus();
   }
   else if (txtMax.Text == "")
   {
      MessageBox.Show("Въведете число!", "Грешка!");
      txtMax.SelectAll(); txtMax.Focus();
   }
```

```
    else
    {
        string a, b, c, d;
        a = btn1.Text;       b = btn2.Text;
        c = btn3.Text;       d = btn4.Text;
        if (string.Compare(a + b, b + a) == 1) { string x = a; a = b; b = x; }
        if (string.Compare(a + c, c + a) == 1) { string x = a; a = c; c = x; }
        if (string.Compare(a + d, d + a) == 1) { string x = a; a = d; d = x; }
        if (string.Compare(b + c, c + b) == 1) { string x = b; b = c; c = x; }
        if (string.Compare(b + d, d + b) == 1) { string x = b; b = d; d = x; }
        if (string.Compare(c + d, d + c) == 1) { string x = c; c = d; d = x; }

        string maxNumber = d + c + b + a;
        string minNumber;
        if (a == "0"&& b == "0"&& c == "0") minNumber = d + a + b + c;
        else if (a == "0"&& b == "0") minNumber = c + d + a + b;
        else if (a == "0") minNumber = b + a + c + d;
        else minNumber = a + b + c + d;

        if (txtMin.Text != minNumber) pcbMin.Image = Properties.Resources.error;
           else pcbMin.Image = Properties.Resources.ok;
        if (txtMax.Text != maxNumber) pcbMax.Image = Properties.Resources.error;
           else pcbMax.Image = Properties.Resources.ok;
        pcbMin.Visible = true;
        pcbMax.Visible = true;
    }
}
private void btnClose_Click(object sender, EventArgs e)    {  Close(); }
```

Задача 7. Различен банков код

Може би си спомняте Петър, който преди време ви помоли да му помогнете с направата на програма за генериране на PIN код. Този път той има нужда от програма, която да генерира четирицифрен PIN код, като всички цифри, участващи в него, са различни.

Изискването за 4 различни цифри налага използването на оператор за цикъл за намиране на поредната цифра, така че тя да не съвпада с вече избраните. Генерирането на различни цифри може да бъде реализирано и по друг начин, посочен в упътването към задача 9.

Първи и втори етап от решението могат да се спестят, защото съвпадат с тези на задача 2.

Решение

Основни цели:

– Прилагане на усвоените знания, умения и навици за:

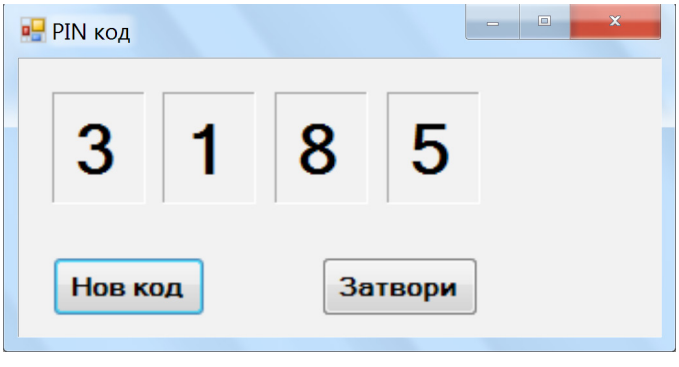
  – проектиране на несложна форма, съдържаща етикети и бутони;

  – настройване в режим на дизайн на основните свойства на използваните контроли;

– именуване на обекти контроли и променливи съгласно общоприета конвенция;
– задаване функционалност на бутон;
– използване на вградени функции за преобразуване на цяло число в низ.

– Изграждане на умения за работа с оператора за цикъл **do-while**.
– Развиване на логическото мислене.
На учениците може да се предложи следното **решение.**

Първи етап – създаване на ГПИ на приложението (фигура 7).



Фигура 7. Варианти на интерфейса

Втори етап – настройка на някои свойства на елементите на ГПИ (таблица 6).

Таблица 6

Елемент на ГПИ	Име	Свойства
Button	btnNew	Text = "Нов код"
Button	btnClose	Text = "Затвори"
Label	lblFirst	Text="" AutoSize=False
Label	lblSecond	Text="" AutoSize=False
Label	lblThird	Text="" AutoSize=False
Label	lblFourth	Text="" AutoSize=False

Трети етап – добавяне на програмен код към елементите.

```
private void btnNew_Click(object sender, EventArgs e)
{
      Random rnd = new Random();
      lbl1.Text = rnd.Next(1, 10).ToString();
      do
      { lbl2.Text = rnd.Next(0, 10).ToString(); }
```

```
        while (lbl1.Text == lbl2.Text);
        do
        { lbl3.Text = rnd.Next(0, 10).ToString(); }
        while (lbl1.Text == lbl3.Text || lbl2.Text == lbl3.Text);
        do
        { lbl4.Text = rnd.Next(0, 10).ToString(); }
        while (lbl1.Text == lbl4.Text || lbl2.Text == lbl4.Text
               || lbl3.Text == lbl4.Text);
}
```

Задача 8. Парола

Васил ежедневно сърфира в интернет пространството и много често се сблъсква с изискването за регистрация. Разбира се, той знае, че дългата парола, включваща букви, символи и знаци, ще го предпази най-добре от злонамерени лица и ще му даде повече сигурност. Желателно е и паролите му да не са едни и същи и затова ви моли да създадете програма, която да генерира такива пароли.

Решение

Основни цели:

– Прилагане на усвоените знания, умения и навици за:

– проектиране на несложна форма, съдържаща етикети и бутони;

– настройване в режим на дизайн на основните свойства на използваните контроли;

– именуване на обекти контроли и променливи съгласно общоприета конвенция;

– задаване функционалност на бутон;

– работа с низове;

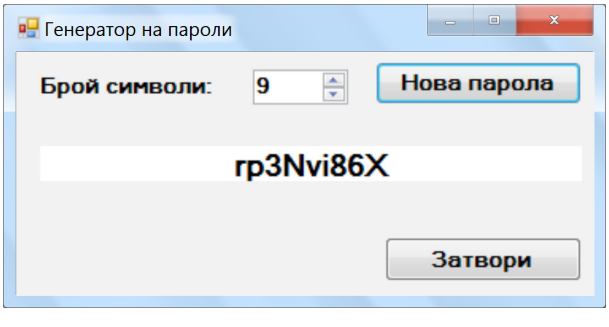
– използване на вградени функции за преобразуване на цяло число в низ.

– Изграждане на умения за работа с **циклични оператори и масиви.**

– Развиване на логическото мислене.

На учениците може да се предложи следното **решение.**

Първи етап – създаване на ГПИ на приложението (фигура 8).



Фигура 8. Варианти на интерфейса

Втори етап – настройка на някои свойства на елементите на ГПИ (таблица 7).

Таблица 7

Контрол	Име на обект	Свойства
Button	btnNew	Text = "Нова парола"
Button	btnClose	Text = "Затвори"
Label	Lbl1	Text= "Брой символи"
TextBox	txtPass	Text= ""
NumericUpDown	UpDownMax	Value=4

Трети етап – добавяне на програмен код към елементите.

В решението е предложена идея, при която предварително в масив от тип `char` (или в променлива от тип `string`) се съхранят всички допустими символи за паролата. За пореден символ, участващ в паролата, се взема този, чийто индекс е случайно число от интервала [0, 61].

I начин *(чрез масив от тип* `char`*):*

```
private void btnNew_Click(object sender, EventArgs e)
{
      char[] letter = newchar[62];
      string pass="";
      int index = 0;
      for (char i = '0'; i <= '9'; i++,index++) letter[index]= i;
      for (char i = 'a'; i <= 'z'; i++,index++) letter[index] = i;
      for (char i = 'A'; i <= 'Z'; i++,index++) letter[index] = i;
      Random rnd = new Random();
      byte countChar= (byte)UpDownMax.Value;
      byte ind;
      string pass="";
      for (byte i = 1; i <= countChar; i++)
      {
                ind = (byte)rnd.Next(0, 62);
                pass = pass + letter[ind];
      }           txtPass.Text = pass;
}
```

II начин *(чрез* `string`*) – аналогично решение само с низове:*

```
private void btnNew_Click(object sender, EventArgs e)
{
      string letter = "";
      //letter - променлива, в която пазим всички позволени символи за паролата
      //добавяме към letter всички цифри,малки и главни букви
      for (char i = '0'; i <= '9'; i++,index++) letter+= i;
      for (char i = 'a'; i <= 'z'; i++) letter += i;
      for (char i = 'A'; i <= 'Z'; i++) letter += i;
      Random rnd = new Random();
      byte countChar = (byte)UpDownMax.Value;
      byte ind;
      string pass = "";// променлива, която ще съдържа генерираната парола
```

```
for (byte i = 0; i < countChar; i++)
{
    //вземаме символ от низа letter със случаен индекс
    ind = (byte)rnd.Next(0, 62);
    pass = pass + letter[ind];
}
txtPass.Text = pass;
```

Следващите две задачи е подходящо да се разгледат в часовете за факултативна подготовка или друга извънкласна форма.

Задача 9. Бикове и крави – класическа игра

Вариант на реализация на играта е представен в (Azalov&Zlatarova, 2011).

Бикове и крави е логическа игра за отгатване на числа. Играе се от двама противници, като всеки се стреми да отгатне тайното число, намислено от другия. Тайните числа са четирицифрени, като цифрите не трябва да се повтарят. След всеки ход противникът отговаря, като посочва броя на съвпаденията – ако дадена цифра от предположението се съдържа в тайното число и се намира на точното място, тя е „бик“, ако пък е на различно място, е „крава“.

Пример: тайно число: 4271.

Предположение: 1234 *Отговор*: „1 бик и 2 крави“.

Да се напише компютърна версия на играта, в която потребителят трябва да познае (най-много с 10 хода) „намислено“ от компютъра число.

Решение

Основни цели:

– Прилагане на усвоените знания, умения и навици за:

– проектиране на форма, съдържаща основните елементи на ГПИ, изучени през учебната година;

– настройване на основните свойства на използваните контроли;

– задаване функционалност на използваните контроли;

– използване на списъчни кутии.

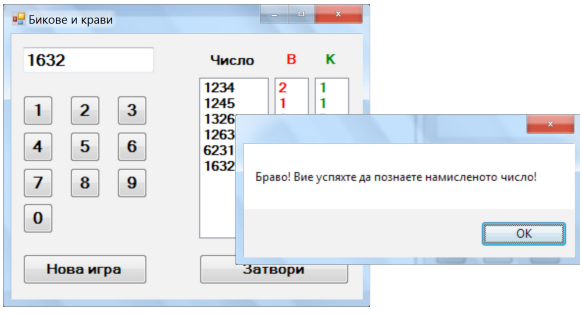
– Разширяване и задълбочаване на практическите и приложните умения на учениците.

– Изграждане на умения за работа с локални и глобални променливи.

– Развитие на логическото мислене.

На учениците може да се предложи следното **решение.**

Първи етап – създаване на ГПИ на приложението (фигура 9).



Фигура 9. Варианти на интерфейса

Втори етап – настройка на някои свойства на елементите на ГПИ (таблица 8)

Поради изискването, че въведеното от потребителя число не може да започва с цифрата нула, тук е предложен ГПИ, при който за въвеждане на числото се използват бутони за всяка цифра. Бутонът с цифра нула става активен едва след въвеждане на поне 1 цифра.

Таблица 8

Контрол	Име на обект	Свойства
Button	btnNewGame	Text = "Нова игра"
Button	btnClose	Text = "Затвори"
Label	LblNumber	Text= "Число"
Label	LblB	Text= "B "
Label	LblC	Text= "K"
Button	Btn0	Text= "0", Enable=False
…		
Button	Btn9	Text= "9"
ListBox	lstNumber	
ListBox	lstB	
ListBox	lstK	

Трети етап – добавяне на програмен код към елементите.

Тук ще изложим само част от решението.

В задача 7 вече генерирахме четирицифрено число с различни цифри. Ето още два алгоритъма за генериране на такова число.

```
int hc, sc, dc, ec;//съответно цифрата на хилядните, стотиците, десетиците и
                   // единиците на ”намисленото” от компютъра число
```

```
//I метод
do
       {
           cmpNumber = p.Next(1022, 9877);
           hc = cmpNumber / 1000;
           sc = cmpNumber / 100 % 10;
           dc = cmpNumber / 10 % 10;
           ec = cmpNumber % 10;
       }
while (hc == sc || hc == dc || hc == ec || sc == dc || sc == ec || dc == ec);

//  II метод
```

Описание:

Декларираме масив a[10], чиито елементи инициализираме с цифрите от 0 до 9.

За първа цифра на числото избираме произволен елемент (цифра) с индекс ind >0 (първата цифра на числото не може да е нула).

Разменяме местата на избрания и последния елемент в масива.

За всяка от следващите цифри:

- генерираме случаен индекс **ind** между 0 и **k**,където
k = 9 – броя на избраните досега цифри.
- стойността на елемента **a[ind]** е поредната цифра на числото.
- разменяме стойностите на a[**ind**] и a[**k**].
- стойността на **k** намалява с 1.

Фрагмент от кода:

```
int[] a = newint[10] { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 };
int ind = p.Next(1, 10);
hc = a[ind];//цифра на хилядните
int x; x = a[ind]; a[ind] = a[9]; a[9] = x;

ind = p.Next(0, 9);
sc = a[ind]; //цифра на стотиците
x = a[ind]; a[ind] = a[8]; a[8] = x;

ind = p.Next(0, 8);
dc = a[ind]; //цифра на десетиците
x = a[ind]; a[ind] = a[7]; a[7] = x;

ind = p.Next(0, 7);
ec = a[ind]; //цифра на единиците
```

Задача 10. Изпитване

Вече помогнахте (в задача 1) на г-жа Петрова с компютърна програма, генерираща случаен номер на ученик от класа, който да бъде изпитан. В част от класовете обаче тя преподава само на едната група, а понякога има и отсъстващи ученици. Напишете програма, чрез която г-жа Петрова ще има възможност да:

– избира начален и краен номер на учениците в класа;
– отбелязва отсъстващите ученици;
– избира на случаен принцип номер на ученик измежду присъстващите, който да бъде изпитан.

Решение

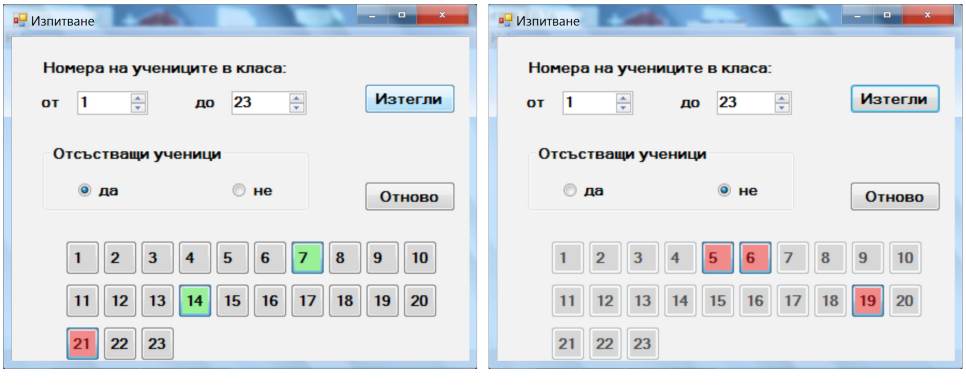
Основни цели:

– Прилагане на усвоените знания, умения и навици за:
  – проектиране на форма, съдържаща основните елементи на ГПИ, изучени през учебната година;
  – настройване в режим на дизайн на основните свойства на използваните контроли;
  – именуване на обекти контроли и променливи съгласно общоприета конвенция;
  – задаване функционалност на използваните контроли.

– Разширяване и задълбочаване на практическите и приложните умения на учениците.
– Разширяване на знанията на учениците за създаване на масив от обекти.
– Изграждане на умения за динамично добавяне и премахване на елемент на ГПИ.
– Изграждане на умения за работа с локални и глобални променливи.
– Развитие на логическото мислене.

На учениците може да се предложи следното **решение.**

Първи етап – създаване на ГПИ на приложението (фигура 10).



Фигура 10. Варианти на интерфейса

Втори етап – настройка на някои свойства на елементите на ГПИ (таблица 9).

Таблица 9

Контрол	Име на обект	Свойства
Button	btnFind	Text = "Изтегли"
Button	btnAgain	Text = "Ново изпитване"
Button	btnClose	Text = "Затвори"
Label	Lbl1	Text= "от №"
Label	Lbl2	Text= "до №"
NumericUpDown	a	Value=1, minimum=1
NumericUpDown	b	Value=35, maximum=35
GroupBox	groupBox1	Text = "Отсъстващи ученици"
RadioButton	rbtnYes	Checked=False
RadioButton	rbtnNo	Checked=True

Трети етап – добавяне на програмен код към елементите.

В решението е предложена идея, при която динамично се създава масив от тип CheckBox. Той служи за визуализиране на номерата на учениците и отбелязване на отсъстващите. При избиране на това кой номер да бъде изпитан, се прави проверка дали този ученик присъства и дали вече не е избран. Номерът на отсъстващия ученик се оцветява в зелен цвят, а на този, който трябва да бъде изпитан, в червен цвят.

Ето фрагмент от кода, чрез който динамично се създава и визуализира масив от CheckBox.

```
CheckBox[] chb = newCheckBox[35];
private void FrmExamp2_Load(object sender, EventArgs e)
{int x, y; //координати на първият CheckBox
   x = 50; y = 150;
   for (int i = 0; i < 35; i++)
   {
       if (i % 10 == 0) { y = y + 40; }
       chb[i] = newCheckBox();
       chb[i].Size = new Size(32, 32);
       chb[i].Appearance = Appearance.Button;
       chb[i].Visible = true;
       chb[i].BackColor = Color.LightGray;
       chb[i].Checked = false;
       chb[i].Enabled = false;
       chb[i].CheckedChanged += newEventHandler(chb_Click);
       this.Controls.Add(chb[i]);
   }
}
```

Заключение

В настоящия материал са включени 10 задачи за формиране на основните компетентности в обучението по информатика съгласно новата учебна програма за VIII клас.

Използването на генератора на случайни числа и игровият модел в тези задачи служат за творческо прилагане на формираните знания и умения за създаване на приложения с ГПИ при решаване на реални житейски задачи и предполага повишаване интереса на учениците към усвояване на нови знания в областта на информатиката и математиката.

Благодарности. Изследването е финансирано от Фонд „Научни изследвания“, проект „Педагогически и технологични аспекти на образователните компютърни игри“.

NOTES/БЕЛЕЖКИ

1. Ministry of Education and Science, General Education, School programs Curricula for the VIII grade, 2017 (in Bulgarian).

REFERENCES/ЛИТЕРАТУРА

Aneva, S. (2010). System of basic problems in learning of event-driven programming with Visual C# Environment in high school. *Proceedings of the scientific Conference "Education in the Information Society"*, (стр. 239 – 251). Plovdiv (in Bulgarian).

Aneva, S. (2010). The role of basic problems in learning of event-driven programming with Visual C# environment in high school. *Proceedings of the Anniversary International Conference "Synergetics and Reflection in Mathematics education"* (стр. 353 – 363). Bachinovo: International Conference "Synergetics and Reflection in Mathematics education" (in Bulgarian).

Aneva, S. (2013). *Model for Professionally Educated Informatics and Information Technology in High School.* Plovdiv: Author's Summary of o.'s Dissertation (in Bulgarian).

Asenova, P. (1989). Izuchenie algoritmicheskoy konstruktsii vibora varianta. *Informatika i obrazovanie, 5*, 45 – 49 (in Russian).

Azalov, P. & Zlatarova, F. (2011). *C++ v primeri, zadachi i prilozhenia.* Sofia: Prosveta. (in Bulgarian)

Dureva, D. (2003). *Problems of the Methodology of Informatics and Information Technology Education.* Blagoevgrad: Neofit Rilski (in Bulgarian).

Garov, K. (2004). A System of Supporting Problems in the Training of Talented Students for Participation in Olympiads and Competitions in Informatics. *Proceedings of the Thirty Third Spring Conference of the Union of Bulgarian Mathematicians, April 1 – 4, 2004, Borovets* (стр. 316 – 321). Sofia: Mathematics and Education in Mathematics (in Bulgarian).

Garov, K. (2010). For the Informatics and Information Technology Training Tasks. Plovdiv: *Procceding of the National Conference "Education in the Information Society"*. Plovdiv, 95 – 101 (in Bulgarian).

Garov, K., & Aneva, S. (2004). The Role of the Problems in Learning of Event-Driven Programming with Visual Basic in High Schools, 2004. *Proceedings of the Thirty Third Spring Conference of the Union of Bulgarian Mathematicians, April 1 – 4, Borovets* (стр. 322 – 328). Sofia: Mathematics and Education in Mathematics (in Bulgarian).

Grozdev, S., & Garov, K. (2008). On the System of Supportive Problems in the Preparation for Participation in Informatics Olympiads Combinatorial Objects and Algorithms. *Proceedings of the Thirty Seventh Spring Conference of the Union of Bulgarian Mathematicians,Borovets* (стр. 304 – 311). Sofia: Mathematics and Education in Mathematics (in Bulgarian).

Grozdev, S., Chehlarova, T., & Terzieva, T. (2010). The need for the development of algorithmic thinking in computer science education. *Proceedings of the scientific Conference "Education in the Information Society"* (стр. 102 – 108). Plovdiv: Development of variational thinking skills in Programming teaching (in Bulgarian).

Hristova, P. & Atanasova, G. (2011). Podgotovka na rakovoditeli na shkoli po informatika. *IV Natsionalna konferentsia "Obrazovanieto v informatsionnoto obshtestvo* (стр. 303 – 310). Plovdiv: IV Natsionalna konferentsia "Obrazovanieto v informatsionnoto obshtestvo" (in Bulgarian).

Manev, K., Maneva, N., & Hristova, V. (2017). *Informatika VIII klas, obshtoobrazovatelna podgotovka.* Sofia: Izkustva (in Bulgarian).

Momcheva, G., Glushkova, T., & Marinova, R. (2017). *Informatika VIII klas.* Sofia: Anubis.(in Bulgarian)

Terzieva, T. (2015). *Creating Graphical User Interface C #.* Plovdiv: Paisii Hilendarski (in Bulgarian).

Yadkova, T., & Lazarova, S. (1989). Reshavaneto na zadachi po informatika. *Obuchenieto po matematika i informatika, V* (2) (in Bulgarian).

Yadkova, T. (1991). *Formirane na metodika za obuchenie v sastavyane na osnovni algoritmi i programi, Doktorska disertatsia.* Blagoevgrad (in Bulgarian).

CREATING OF GAMES IN THE INFORMATICS CLASSES BY USING A GENERATOR OF RANDOM NUMBERS

Abstract. Methodological ideas are given in the paper for using the random numbers generator in programming education. A set of problems is presented in connection with visual programming education on the base of the C# language. The aims and the main stages are exposed for the solution of each problem. Also, the problems illustrate a possibility to combine the process of programming education with the creation of games. The set under consideration contains 10 problems, including 7 ones which are original and 3 ones are well-known classic games. The aim and the main stages of the solution are presented for each problem.

✉ **Ms. Emilia Nikolova, PhD student**
Department of Mathematics
South-West University "Neofit Rilski"
66, Ivan Mihailov St.
2700 Blagoevgrad, Bulgaria
E-mail: e_nikol@abv.bg

✉ **Dr. Daniela Tuparova, Assoc. Prof.**
Department of Informatics
South-West University "Neofit Rilski"
66, Ivan Mihailov St.
2700 Blagoevgrad, Bulgaria
E-mail: ddureva@swu.bg