Educational Technologies
Образователни технологии

МЕТОДИ ЗА АКТИВНО ОБУЧЕНИЕ НА КАНДИДАТ-СТУДЕНТИ ПО ИНФОРМАТИКА

Христо Крушков
Пловдивски университет „Паисий Хилендарски“

Резюме. Потребността от висококвалифицирани специалисти в областта на информационните и комуникационните технологии нараства с всяка изминала година. Отчитайки тази тенденция, МОН предприе мерки за стимулиране на висшите учебни заведения, които имат необходимия капацитет да увеличат приема на студенти в професионално направление 4.6 „Информатика и компютърни науки“, а така също и в направление 4.5 „Математика“. Математиката и информатиката са дисциплини, които затрудняват голяма част от учениците, и мотивацията за усвояване на знания и умения, която преподавателят може да стимулира, е от особена важност.

В статията е описан опитът на автора при обучение на кандидат-студенти в подготовката им за конкурсен изпит по информатика в Пловдивския университет „Паисий Хилендарски“. Подходи за повишаване на мотивацията им като обучение чрез забавление (edutainment) и игровизация дават добри резултати. Предложен е и метод за решаване на задачи за кандидатстудентски изпит и е представено примерно решение на конкретна задача.

Keywords: computer science education, active learning, teaching methodology

Въведение

Недостигът на квалифицирани кадри в областта на информационните и компютърните технологии (ИКТ) става все по-осезаем. Естественото място за подготовка на такива кадри са ВУЗ. Колкото и да се разширява капацитетът на тези заведения за обучение на студенти в направлението, той все пак е лимитиран. На пазара за образователни услуги се появяват фирми и организации, които не са акредитирани от Националната агенция за оценяване и акредитация (НАОА). Отчитайки тази тенденция, МОН предприе мерки за стимулиране на ВУЗ, които имат необходимия капацитет, да увеличат приема на студенти в професионално направление 4.6 „Информатика и компютърни науки“, а така също и в направление 4.5 „Математика“. Остава въпросът дали местата в тези направления ще се запълнят от студенти с адекватно входно ниво, а има даже и опасения, че може да останат и незаети места. Математи-

ката и информатиката са дисциплини, които затрудняват голяма част от учениците, и мотивацията за усвояване на знания и умения, която преподавателят може да стимулира, е от особена важност.

Кандидатстудентски изпит по информатика се провежда в Пловдивския университет „Паисий Хилендарски“ от 2003 година. От същата година започва провеждане на курсове за подготовка на кандидат-студенти за явяване на този изпит. Учебната дисциплина „Информатика“ е залегнала в учебните планове за средно образование. Теоретично, завършващите средношколци трябва да имат всички необходими знания за решаване на задачата от кандидатстудентския изпит, защото той е съобразен с Държавните образователни изисквания за учебно съдържание по „Информатика и информационни технологии“ (2000, 13.06.). Редица учебници способстват за подготовката на кандидат-студентите (Azalov & Zlatarova, 2001), (Barnev et al., 2001), (Manev & Maneva, 2002). На практика, без допълнително обучение успяват да се справят предимно завършилите специализирани гимназии или паралелки. За да могат да съставят програма по задачата, е необходимо да познават основни управляващи конструкции, структури от данни и алгоритми за сортиране, търсене, филтриране, намиране на суми, средноаритметични, минимални и максимални елементи, текстообработка. Създаването и извикването на подпрограми е от съществено значение за добро решение. Тези, които идват на кандидатстудентски курс, не притежават описаните знания в степен да могат да решат конкурсна задача. Трябва да ги получат в рамките на четири месеца.

Методи за активно обучение на кандидат-студенти

Първата основна цел при обучението на кандидат-студентите е свързана с намаляването на абстракцията, която съществува при моделирането и реализирането на структурите от данни и алгоритмите, които ги обработват. Подходящи средства в тази посока са визуалните алгоритми (Grozdev & Terzieva, 2011). С тяхна помощ обучаемите могат да получат много добра представа за изпълнението на даден алгоритъм, както и да го проследят стъпка по стъпка. Дава им се възможност да експериментират с различни методи за сортиране и търсене, както и да сравняват тяхната ефективност (Grozdev & Terzieva, 2012). Намирането на минимален, максимален елемент, суми и средноаритметични започват да стават все по-понятни. Конструкциите за повторение се изясняват в интерактивна среда (Krushkova, 2014). Обучението се превръща в забавление.

Втората цел е свързана с интензификацията на обучението. За да могат да бъдат усвоени всички необходими знания за краткия период, изключително важно е курсистите да бъдат стимулирани да участват активно в подготовката, предлагайки самостоятелни решения на поставените задачи за домашна работа. В това отношение спомага въвеждането на състезателен характер на

обучението. Изпратените решения се оценяват по начина на точкуването на кандидатстудентски изпит, след което се прави класиране, което се вижда от всички курсисти. Ползата от този подход се проявява още на второто класиране. Тези, които подхождат небрежно, започват да стават по-акуратни и решенията им се подобряват. За стимулиране на процеса по непрекъснато подобряване на уменията до края на втория месец курсистите имат право на три „опита“. Това означава, че след посочени от преподавателя грешки при поредния „опит“ могат да пратят коригирано решение. В класирането се участва с точките от последния „опит“. Третия месец се разрешават два опита, а през последния с цел оптимална подготовка за самия изпит – само един „опит“.

Предлагането на подходящ модел за решаване на задача от конкурсен изпит е третата важна предпоставка за успешно представяне на изпита. При проектиране на решение кандидат-студентите могат да го ползват, работейки по метода на аналогията. Ще продължим с индуктивно извеждане на модела, като е решена една конкретна задача, след което е описан и самият модел.

Задача от конкурсен изпит по информатика в Пловдивския университет „Паисий Хилендарски“

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.

Указание към задачата

При решаване на задачата по програмиране трябва:

а) да се опише словесно използваният алгоритъм;

б) да се опише на хартия решението на задачата на един от следните езици за програмиране: **Pascal, C, C++, C#, Java, Basic**;

в) да се коментира написаният текст на програмата, като се посочи предназначението на основните променливи и структури от данни и използваните процедури и функции.

Анализ на задачата

Тази задача позволява да се демонстрира изключителната полза от усвояване на знания за работа с подпрограми. Въпреки че обектноориентираният стил на програмиране се наложи като водещ в последните години, голяма част от кандидат-студентите, които са изучавали програмиране, не са обучени нито за работа с класове и обекти, нито с подпрограми. За преодоляване на този пропуск и усвояване на механизма за предаване на параметри е създаден специален визуален алгоритъм (Krushkova et al., 2010). При представянето на решение се стремим да предложим максимално опростен и стандартизиран интерфейс за комуникация с подпрограмите.

При първи прочит на условието става ясна основната структура **student**, която трябва да съдържа три текстови полета с конкретизирана максимална дължина и едно целочислено. При дефинирането 3 добавяме към дължината на всяко поле 1, за да се запази място и за край на низ (‘\0‘). При внимателен повторен прочит се вижда, че на два пъти се използва модифициран текст с кратко име на преподавател, в който фигурират инициали. За удобство добавяме пето поле, в което поставяме краткото име. Дефинираме и масив от студенти с име **Tmasiv**. Пак с цел удобство при работа увеличаваме максималния брой елементи на масива N=40 с единица, за да работим с индекси от 1 до 40.

Функцията **instudent** служи за въвеждане на данни за един студент. Формалният параметър е псевдоним, за да може подаденият фактически параметър да получи попълнената с данни структура. Стойностите на четирите полета се въвеждат от клавиатурата. Петото поле се изчислява автоматично. При получаването му се предполага, че полето с името на преподавателя е коректно въведено. Допълнителни проверки за коректност не се изискват в условието, а биха затруднили обучаемите, ако ги добавим в програмния текст. В името на преподавателя към интервалите се насочват два указателя s1 и s2: char *s1=strchr(a.prep,' '), *s2=strrchr(a.prep,' ');

				s1							s2					
Е	м	и	л		П	е	т	р	о	в		Г	о	ц	е	в

Преместването им надясно става посредством s1++; s2++;

					s1							s2				
Е	м	и	л		П	е	т	р	о	в		Г	о	ц	е	в

След това strcpy(a.iniprep,” . . “); запълва полето **iniprep** с шаблон за инициали

0	1	2	3	4	5	6	7	8	9							
	.			.												

a.iniprep[0]=a.prep[0]; // Попълва първа позиция (с индекс 0) на шаблона

0	1	2	3	4	5	6	7	8	9							
Е	.			.												

a.iniprep[3]=s1[0]; // Попълва четвърта позиция (с индекс 3) на шаблона

0	1	2	3	4	5	6	7	8	9							
Е	.		П	.												

strcat(a.iniprep,s2); // Долепя отдясно на шаблона фамилията

0	1	2	3	4	5	6	7	8	9							
Е	.		П	.		Г	о	ц	е	в						

По този начин след приключване на изпълнението си функцията е заредила със стойности четири полета от клавиатурата, а петото е попълнила автоматично. Наличието на това пето поле изключително опростява следващата функция, която отпечатва данни за един студент – **outstudent**. В нея просто се извеждат първите три полета и последното.

Следващите две функции **input** и **output** служат съответно за въвеждане на масив от студенти и отпечатване на масив от студенти. Параметърът **a** е масивът, а **br** е действителният брой на студентите в групата. В тези, а и в следващите функции, нулевият елемент на масива не се използва. Управляващата променлива пробягва индексите от 1 до **br**. Така обработката е по-естествена и разбираема за обучаемите.

За да изпълним точка 2 от условието, трябва да реализираме функция за сортиране по факултетен номер – **sortfnom**. Използваме стандартен метод на мехурчето, програмният текст на който е подходящ за начинаещи. При тази сортировка, а и при следващите функции, параметрите са същите – масив и действителен брой елементи. Вече имаме всичко необходимо за решаване на задачата до точка 2: с **input** въвеждаме група студенти, със **sortfnom** я сортираме по факултетен номер, с **output** я отпечатваме на екрана.

Две функции са необходими за решаване на трета точка. Едната – **sortoc,** чрез метода на мехурчето реализира сортировка по два критерия, включваща следното съставно условие: ако двама съседни студенти са неправилно подредени по успех (предният е с по-нисък успех от следващия) или двама съседни студенти имат равен успех и са неправилно подредени по азбучен ред (предният не е азбучно преди следващия), то им се разменят местата. Другата функция – **spravka2**, филтрира записите и отпечатва само тези, които отговарят на условието да съдържат в името си „Димо“ или „димо“. Последователното им извикване решава тази точка.

Преди да преминем към главната функция, ще създадем такава, която решава т. 4.б). След улеснението, което създадохме посредством добавянето на изчислимото поле, програмният текст на функцията **spravka3** е значително опростен. Намирането на средноаритметично на група записи, отговарящи на определено условие (петото поле да има стойност „Е. П. Гоцев“), се свежда до преброяването им и сумирането им по полето оценка. Резултатът от функцията е частното на сумата и броя, а ако няма такъв преподавател – 0.

Накрая в главната функция се декларират три масива a, b и c, моделиращи три групи съответно с br1, br2 и br3 броя студенти. При въвеждането на всеки брой е необходимо да бъде контролирана коректността на въведеното число (между 1 и 40). За всяка група се въвеждат данни и изпълняват функциите по точки 2 и 3. Накрая в три променливи s1, s2 и s3 се съхраняват резултатите от **spravka3** за всяка група и се намира техният минимум.

Решение на задачата

Представяме програмния текст на задачата, обстойно коментиран. Програмата е тествана с Dev-C++.

```
#include <iostream>
#include <string.h>
#include <windows.h>
using namespace std;
const
    int N=40,  //макс. брой студенти;
        L=40;  //макс. дължина на име;
struct student
{
        char fnom[11]; // факултетен номер
        char ime[L+1]; // име на студент
        int ocenka;    // оценка
        char prep[L+1],// име на преподавател
        iniprep[L];    // име на преподавател с инициали
};
typedef student Tmasiv[N+1];
void instudent(student &a) // Въвеждане на данни за един студент
{
```

```
        cout<<Въведете ф.номер:»; cin>>a.fnom; cin.ignore();
        cout<<”Въведете име на студент:”; cin.getline(a.ime,L);
        cout<<”Въведете оценка:”; cin>>a.ocenka; cin.ignore();
        cout<<”Въведете име на преподавател:”; cin.getline(a.prep,L);
        char *s1=strchr(a.prep,' '), // Указател към първи интервал
        *s2=strrchr(a.prep,' '); //Указател към последен интервал
        s1++; // s1 се премества позиция надясно и сочи второ име
        s2++; // s2 се премества позиция надясно и сочи фамилия
        strcpy(a.iniprep,” .  . ”); // Шаблон за инициали
        a.iniprep[0]=a.prep[0]; // Попълва първа позиция на шаблона
        a.iniprep[3]=s1[0]; // Попълва четвърта позиция на шаблона
        strcat(a.iniprep,s2); // Долепя отдясно на шаблона фамилията
} //instudent
void outstudent(student a) // Извеждане на данни за един студент
{
     cout<<a.fnom<<”, ”<<a.ime<<”, ”<<a.ocenka<<”, ”<<a. iniprep<<endl;
} //outstudent
void input(Tmasiv a, int br) // Въвеждане на масив от студенти
{
    for (int i=1; i<=br; i++)
    {
         cout<<”Въведете данни за ”<<i<<”-ия студент”<<endl;
      instudent(a[i]);
       }
}//input
void output(Tmasiv a, int br) // Извеждане на масив от студенти
{
      for (int i=1;i<=br;i++)
         outstudent(a[i]);
}//output}
void sortfnom(Tmasiv a, int br) // Възходяща сортировка по ф.номер
{
   for (int i=2; i<=br; i++)
     for (int j=br; j>=i; j--)
      if (strcmp(a[j-1].fnom,a[j].fnom)>0)
          {
                student pom=a[j];
                a[j]=a[j-1];
                a[j-1]=pom;
          }
}//sortfnom
void sortoc(Tmasiv a, int br) // Низходяща сортировка по успех
{          // При еднакъв успех, възходяща по име на студент
    for (int i=2; i<=br; i++)
     for (int j=br; j>=i; j--)
      if (a[j-1].ocenka<a[j].ocenka||
          a[j-1].ocenka==a[j].ocenka&&
             strcmp(a[j-1].ime,a[j].ime)>0)
          {
```

```
            student pom=a[j];
            a[j]=a[j-1];
            a[j-1]=pom;
         }
}//sortoc
void spravka2(Tmasiv a, int br) // Извеждане на студенти с Димо/димо
{
     for (int i=1;i<=br;i++)
      if (strstr(a[i].ime,"Димо")||strstr(a[i].ime,"димо"))
            outstudent(a[i]);
}//spravka2
double spravka3(Tmasiv a, int br) //Изчисляване на среден успех при
{                                 //      «Е. П. Гоцев»
     int count=0;double sum=0;
    for (int i=1;i<=br;i++)
    {
       if (strcmp(a[i].iniprep,»Е. П. Гоцев»)==0)
         { sum+=a[i].ocenka; count++;}
    }
    if (count>0) return sum/count; else return 0;
}//spravka3
int main()
{
  SetConsoleCP(1251); //За да може да работите на кирилица в конзолата
  SetConsoleOutputCP(1251); // използвайте шрифт Lucida Console
  Tmasiv a,b,c; int br1,br2,br3;
  do
    {
    cout<<"Въведете брой студенти в I група:";
           cin>>br1;
      }
  while (br1<0||br1>40);
  cin.ignore();
  input(a,br1);
  cout<<"Списък на I група подреден по ф.номер"<<endl;
  sortfnom(a,br1);
  output(a,br1);
  cout<<"Студенти от I група с Димо/димо в името"<<endl;
  sortoc(a,br1);
  spravka2(a,br1); cout<<endl;
  do
    {
    cout<<"Въведете брой студенти във II група:";
           cin>>br2;
      }
  while (br2<0||br2>40);
  cin.ignore();
```

```
  input(b,br2);
  cout<<"Списък на II група подреден по ф.номер"<<endl;
  sortfnom(b,br2);
  output(b,br2);
  cout<<"Студенти от II група с Димо/димо в името"<<endl;
  sortoc(b,br2);
  spravka2(b,br2); cout<<endl;
  do
    {
    cout<<"Въведете брой студенти в III група:";
          cin>>br3;
       }
  while (br3<0||br3>40);
  cin.ignore();
  input(c,br3);
  cout<<"Списък на III група подреден по ф.номер"<<endl;
  sortfnom(c,br3);
  output(c,br3);
  cout<<"Студенти от III група с Димо/димо в името"<<endl;
  sortoc(c,br3);
  spravka2(c,br3); cout<<endl;
  double sr1=spravka3(a,br1);
  cout<<"Ср.успех при Е. П. Гоцев на I група:"<<sr1<<endl;
  double sr2=spravka3(b,br2);
  cout<<"Ср.успех при Е. П. Гоцев на II група:"<<sr2<<endl;
  double sr3=spravka3(c,br3);
  cout<<"Ср.успех при Е. П. Гоцев на III група:"<<sr3<<endl;
  double min=sr1;
  if (sr2<min) min=sr2;
  if (sr3<min) min=sr3;
  cout<<"Минимален ср.успех при Е. П. Гоцев:"<<min<<endl;
}
```

Обобщен метод за решаване на задача от кандидатстудентски изпит по информатика в Пловдивския университет „Паисий Хилендарски“

Ще представим общата структура на една задача от конкурсен изпит по информатика и ще наблегнем на някои типични подпрограми (ППГ). Преди дефиниране на структурата е необходимо няколко пъти да се прочете условието на задачата. Често се случва освен изрично зададените полета да е удачно, с цел улеснение при по-нататъшното писане на програмен текст, да се добави т.нар. изчислимо поле. Въпреки че това води до заемане на повече памет, може да спести допълнителни обработки в други функции. Пример за такова поле в нашето решение е полето с името на преподавател, съдържащо инициали – **iniprep**. То се използва на два пъти – веднъж при извеждане на информация за студент и втори път при **spravka3**.

1. Дефиниране на структура:
   а) полета, въведени от клавиатура;
   б) полета, изчислими от други полета;
   в) типове на полета:
   – числени;
   – текстови;
   – явно изброим тип;
   – друга структура;
   – масив.
2. Дефиниране на масив от структури.
3. ППГ за въвеждане на един елемент:
   а) въвеждат се полета от клавиатура;
   б) изчисляват се изчислимите полета.
4. ППГ за извеждане на един елемент:
   а) обработка на полета за изход;
   б) извеждане резултатите на екрана.
5. ППГ за въвеждане на целия масив.
6. ППГ за извеждане на целия масив.
7. Извеждане на справки с условия:

```
<тип> spravka(Tmasiv a, int br)
{
  условията са:
  а) константи;
  б) променливи, които се въвеждат от клавиатурата;
  в) променливи, които се изчисляват автоматично.
  for (int i=1; i<=br;i++)
  if (<a[i] удовлетворява условието на справката>)
     outstructura(a[i]);
}
```

8. Сортировки:

```
void sort(Tmasiv a, int br)
  {for (int i=2; i<=br; i++)
    for (int j=br; j>=i; j--)
     if (<a[j-1] и a[j] са неправилно подредени>)
     {structura x=a[j-1]; a[j-1]=a[j]; a[j]=x;}//сменят си местата
  }//sort
```

9. Главна функция.

Ще обърнем внимание на функцията **spravka2**. Нейната структура е модел за много други справки. Съдържа задължителен цикъл за обхождане на всички елементи на масива от структури:

```
for (int i=1; i<=br; i++)
if (<a[i] удовлетворява условието на справката>)
  outstructura(a[i])
```

В цикъла се проверява дали поредният елемент a[i] удовлетворява условието на справката. Условието може да бъде просто или съставно. В примера е съставно (да съдържа „Димо“ или „димо“). Ако условието е истина, се извиква функцията за извеждане данните за една структура на екрана (в горния шаблон е наречена с обобщеното име **outstructura**). Фактически параметър на тази функция е a[i]. Критериите за сравнение могат да бъдат:

а) константи (в примера **„Димо“**);

б) променливи, които се въвеждат от клавиатура (например да се изведат всички студенти, чийто успех надхвърля зададен минимален успех **MIN** – стойността на променливата **MIN** се въвежда от клавиатурата);

в) променливи, които се изчисляват автоматично като функция на всички елементи в масива (например да се изведат всички студенти, чийто успех надхвърля средния успех **S** за целия масив от студенти – стойността на променливата **S** се изчислява автоматично).

Критериите за сравнение от точки б) и в) трябва да се въведат/изчислят преди описания по-горе цикъл.

При някои справки освен филтрирането на елементите се изчисляват т.нар. агрегатни функции (сума, брой, средноаритметично, минимум, максимум и др.). В такъв случай е добре справката да връща резултат от типа на съответната агрегатна функция. В повечето случаи удачните типове са double или int. В примерната задача в **spravka3** се изисква като резултат да се върне средноаритметичното на елементи, които отговарят на конкретно условие. Затова типът е double. И в този случай се вижда улеснението, което предлага изчислимото поле **iniprep**.

Сортировката, която курсистите масово избират, е методът на мехурчето. Въпреки че той е неефективен от гледна точка на бързодействие, в конкретния случай има две предимства. Първото е краткият програмен текст. Второто е, че това е устойчив метод, което ще рече, че при сортиране по определен критерий елементите с еднакви стойности по този критерий запазват подредбата си един спрямо друг такава, каквато е била преди сортирането. Така например, ако искаме да сортираме по среден успех, а тези с еднакъв среден успех – азбучно, можем да приложим метода по последния критерий (азбучна сортировка). След това да го приложим и върху основния критерий (сортировка по успех).

Внимателният читател навярно е забелязал, че в главната функция има повтарящи се фрагменти, които биха могли да се обособят в самостоятелна функция. Това би довело до съкращаване на програмния текст. Тази задача оставяме за самостоятелна работа.

Заключение

Методите за активно обучение са полезни при интензивните кандидатстудентски курсове. За краткия четиримесечен период редовно 30% – 40% от курсистите успяват да усвоят добре материала и постигат отлични оценки още на предварителния изпит. Така те получават правото веднага да бъдат записани в желаната специалност. В последните години във ФМИ на ПУ „Паисий Хилендарски“ най-голям е интересът към новата специалност „Софтуерни технологии и дизайн“. Осъвременяването с помощта на европейски проект на учебните планове и програми на двете по-стари специалности „Бизнес информационни технологии“ и „Информатика“ повиши значително интереса и към тях. Това стана причина част от кандидат-студентите да не успеят да запишат специалност от професионално направление 4.6 „Информатика и компютърни науки“. През настоящата година приемът в това направление е увеличен със 100 броя и се предвижда процесът на повишаване на капацитета да е постоянен.

На сайта на ФМИ при Пловдивския университет „Паисий Хилендарски“ могат да се видят всички конкурсни задачи: http://fmi-plovdiv.org/index.jsp?id=324&ln=1. Основният учебник, който се ползва при подготовката за изпит, е „Програмиране на C++“ (Крушков, 2010). Задачите от него, но решени на Паскал, се предлагат в друго ръководство (Крушков & Илиев, 2007). Материалите от учебниците са достъпни на сайта 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.

REFERENCES / ЛИТЕРАТУРА

Azalov, P. & Zlatarova F. (2001). *Informatika za XI klas*. Sofia: Prosveta. [Азълов, П. & Златарова Ф. (2001). *Информатика за XI клас*. София: Просвета.]

Barnev, P., Totkov, G., Doneva, R., Shkurtov, Vl. & Garov, K. (2001). *Informatika+, uchebnik za profilirana podgotovka v XI klas na SOU*. Plovdiv: Letera. [Бърнев, П., Тотков, Г., Донева, Р., Шкуртов, Вл. & Гъров, К. (2001). *Информатика+, учебник за профилирана подготовка в XI клас на СОУ*. Пловдив: Летера.]

Grozdev, S. & Terzieva, T. (2011). Vizualizatsiya metodov sortirovki massivov. *Elektronnayy zhurnal Rossiyskoy Akademii Obrazovaniya "Informatsionnaya sreda obrazovaniya i nauki", 5*. [Гроздев, С. & Терзиева, Т. (2011). Визуализация методов сортировки массивов.

Электронный журнал Российской Академии Образования „Информационная среда образования и науки“, *5*.]

Grozdev, S. & Terzieva, T. (2012). Statichnaye i dinamichnaye sredstva dlya vizualizatsii metodov sortirovki masivov. *Pedagogicheskaya informatika* (Vipusk 1). [Гроздев, С. & Терзиева, Т. (2012). Статичные и динамичные средства для визуализации методов сортировки масивов. *Педагогическая информатика* (Выпуск 1).]

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.]

Krushkov, Hr. & Iliev, A. (2007). *Programirane na Paskal. Chast I i II (shesto dopalneno i preraboteno izdanie)*. Plovdiv: Universitetsko izdatelstvo “Paisiy Hilendarski”. [Крушков, Хр. & Илиев, А. (2007). *Програмиране на Паскал. Част I и II (шесто допълнено и преработено издание)*. Пловдив: Университетско издателство „Паисий Хилендарски“.]

Krushkov, Hr. (2010). *Programirane na C++. Chast I – Vavedenie v programiraneto (treto dopalneno i preraboteno izdanie)*. Plovdiv, Koala Pres. [Крушков, Хр. (2010). *Програмиране на C++. Част I – Въведение в програмирането (трето допълнено и преработено издание)*. Пловдив, Коала Прес.]

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

Manev, K. & Maneva, N. (2002). *Informatika XI Profilirana podgotovka sas C.* Sofia: Anubis. [Манев, К. & Манева, Н. (2002). *Информатика XI Профилирана подготовка със C*. София: Анубис.]

Krushkova, M., Stoynova, Y. & Krushkov, Hr. (2010). Teaching Subroutines: as Early as Possible. *Anniversary International Conference REMIA 2010*, (pp. 451 – 457). Plovdiv.

METHODS FOR ACTIVE LEARNING OF PROSPECTIVE STUDENTS IN INFORMATICS

Abstract. The need for highly qualified specialists in the field of Information and Communication technologies is growing every year. Recognizing this trend, the Ministry of Education and Science has taken measures to boost higher education institutions that have the capacity to increase the intake of students in the professional direction 4.6 Informatics and Computer Science, as well as in the direction 4.5 Mathematics. Mathematics and computer science are disciplines that hamper the majority of students and it is essential that the teacher can stimulate motivation for learning.

The article describes the experience of the author in training prospective students to prepare for the Informatics entrance exam in Plovdiv University. Approaches to increase motivation as education through entertainment (edutainment) and gamification give good results. A method is proposed for solving problems of entrance exam and an exemplified solutions to a concrete task is described.

✉ **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