TEACHING PROGRAMMING THROUGH GAMES DEVELOPMENT

TEODOSI, K. TEODOSIEV, GALINA Y. TEODOSIEVA

ABSTRACT: *This article suggests developing computer games in the educational process as a possible method to increase students' motivation. Moreover, reviewing language and algorithmic elements can be done though interesting exercises – games. This method illustrates that games can be developed with only starting programming knowledge. Students assume the role of game developers as they practice a real profession. Using a few handpicked game exercises there can be a discussion of how games can be developed while learning main structures and elements of programming. Developing computer games during the educational process may spark interest in new topics and stimulate learning.*

KEYWORDS: *teaching, programming, game, development.*

ПРЕПОДАВАНЕ НА ПРОГРАМИРАНЕ ЧРЕЗ РАЗРАБОТКА НА ИГРИ*

ТЕОДОСИ К. ТЕОДОСИЕВ, ГАЛИНА Й. ТЕОДОСИЕВА

АБСТРАКТ: *Статията предлага разработването на игри в процеса на обучение като възможен подход за повишаване на мотивацията на обучаемите. Става въпрос за разглеждане на езиковите и алгоритмичните елементи чрез интересни задачи – игри. Показва се, че може да се програмират игри, дори с начални програмистки знания. Обучаемите поемат ролята на разработчици на игри, което им дава възможност да практикуват професионална дейност много приличаща на реална. На базата на няколко подбрани игрови задачи се коментира как паралелно с разглеждането на основни програмни конструкции и структури може да се разработват игри. Разработването на компютърни игри по време на обучението може да събуди интерес към нови теми да стимулира запомнянето на вече изученото.*

1 Въведение

В последните години се засили тенденцията към включване на игри в обучението не само на малките ученици, но и на тези в гимназиална степен. Това е обусловено от една страна от липсата на мотивация за учене у младите хора, а от друга от огромния интерес у младежите към компютърните игри. Съвсем наскоро Световната здравна организация призна за болест пристрастяването към компютърните и видео игри. Очевидно е примамливо този интерес да се използва за мотивиране на ученето и повишаване на резултатите от обучението. Съществува и една опасност, чрез игрите да внушим на обучаемите, че ученето е игра, което съвсем не е така. Обучението изисква целенасочени усилия и неминуемо е свързано и с трудности.

2 Игрите в обучението по програмиране

Курсовете по програмиране са необходими за обучаемите, защото са базови за разработката на приложения в бъдещата им дейност. В общи линии целта на тези курсове

* Настоящата статия е частично финансирана по проект № РД-08-122/6.02.2018

е да позволят на обучаемите да овладеят знанията: представяне на задачата в набор от точно определени стъпки, а след това да ги кодират на език за програмиране.

Обучението по програмиране се сблъсква със сериозни проблеми. Много обучаеми се отказват от програмирането, защото не могат да разберат понятията, които им се преподават в курса „Увод в програмирането". Сред причините за този проблем са: ниска мотивация, отсъствие на способност за абстрактно мислене, ниско ниво на математически знания [4].

Тези трудности водят при много обучаеми до затруднения при усвояването на материята и до загуба на интерес към програмирането. От друга страна разработването на игри е възможност за обучаемите да изпълнят фактическа професионална дейност и следователно може да служи за стимул за тяхната мотивация и активност.

Изборът на задачи в уводния курс по програмиране е тежка задача. От една страна задачата да е лека за разбиране, за да се концентрира вниманието на обучаемите върху алгоритмичните и езикови елементи. От друга страна е добре, задачата да е интересна, за да повиши мотивацията им. Тук обаче се появяват проблемите с непознаването на предметната област и съставяне на модела. В този смисъл игрите изглеждат добър избор. Игрите са познати на обучаемите (много от тях са ги играли на хартия).

Алтернатива за генериране на успех в процеса на обучение се явява използването на игри, тъй като те могат да прибавят в образователния контекст преимущества като мотивация, наслада, интердисциплинарност и развитие на когнитивните способности. Могат да се използват два начина за включване на компютърните игри в обучението.

Първият е известен като „сериозни игри", в които обучаемите участват като играчи, т.е. те трябва да решават задачата колективно или индивидуално. Vahldick et al. [5] представя в своя обзорна статия 40 игри, свързани с предмета и компетенциите във уводния курс по програмиране. Идеята се заключава в това да се предоставят на обучаемите игри, т.е обучаемите се срещат с предварително поставени проблеми и трябва да ги преодолеят.

Има и друг начин за използване на игрите. Като се използва интереса към игрите, разработването на игри се включва по време на обучението. В този случай се разглеждат езиковите и алгоритмични елементи чрез интересни задачи. Показва се, че може да се програмират игри, дори с начални програмистки знания. При втория начин обучаемите поемат ролята на разработчици на игри, което им дава възможност да практикуват професионална дейност, много приличаща на реална.

3 Набор от игри

Решаването на задачи е задължителен елемент на съдържанието на обучението по програмиране. Решавайки задачи, обучаемите овладяват умения и навици за използването на теоретичните знания в практиката. Нещо повече, именно способността за решаване на задачи, т.е. извършване на определени действия с информацията от условието на задачата, означава овладяване на знания.

Разработката на игри през уводния курс по програмиране предоставя на обучаемите опит за всекидневна работа, която ще практикуват през кариерата си. Позитивните и проактивни отношения по време на разработката на играта са очевидни особено по отношение на търсене на нови знания.

На базата на няколко подбрани игрови задачи се коментира как паралелно с разглеждането на основни програмни конструкции и структури може да се разработват

игри. По тази причина препоръките за избор на задачите са показани и коментирани като част от всяка от основните теми в курса по “Увод в програмирането”.

3.1 Скаларни типове данни. Оператор за присвояване. Операции за отношение. Условни оператори.

Може да се започне с игра подходяща за малки ученици.

✓ *"Калкулатор"[1]. Проста обучаваща програма, използваща "вродените" способности на компютъра – сметачните. Програмата генерира две псевдослучайни числа от зададен интервал и избира аритметична операция(+,-*), след което дава възможност за отговор на ученика и проверява отговора.*

Това е обучение или упражнение в игрова обстановка. Целта е придобиване на някакъв полезен навик или знание.

```
#include<iostream>
#include<stdlib.h>
using namespace std;
int main()
{ int arg1,arg2,ans,rez,k;char op;
 system("chcp 1251");
 srand(time(0));arg1=rand()%1001;//1
 arg2=rand()%1001;
 k=rand()%3;
 op= (0==k)? '+' : (1==k)? '-' : '*';
 cout<<"Компютърът генерира две цели числа в интервала [0, 1000] "<<endl;
 cout<<"и аритметична операция + - * "<<endl;
 cout<<arg1<<op<<arg2<<"="<<endl;
 cout<<"Въведи резултата: ";cin>>ans;
 rez= ('+'==op)? arg1+arg2 : ('-'==op)? arg1-arg2 : arg1*arg2;//2
 if (ans==rez) cout<<"Вярно!"<<endl;
 else cout<<"Грешка!Верният резултат е  "<<rez<<endl;
 return 0;
}
```

Тук може да се запознаят обучаемите с функциите за генериране на псевдослучайно число. Тъй като функцията rand() генерира число от 0 до rand_max, трябва да се покаже как да се преформатира в цяло число в [a,b] (//1). Тази задача дава възможност и за упражняване на семантиката на условната операция (//2). При разглеждане на управляващите конструкции най-важният момент е правилното конструиране на условията (булеви изрази) и правилното им използване.

В езиците, които използват един символ (=) за присвояване и друг (==) за сравнение (например C/C++), когато сравненията могат да бъдат направени в рамките на структурите за контрол, предимство е мястото на константи или изрази да е вляво на всяко сравнение. Изразите с ляво сравнение **('+'==op)** и **('-'==op)** биха предизвикали грешка при компилация при пропускане на едното =, което е обичаен пропуск при начинаещите. Това няма да се случи при дясното (стандартно) сравнение.

След въвеждане на операциите с цели и реални числа и разклоняване на програми, може да се реши следната задача.

✓ *"Календар". Тази програма по въведена дата за произволно зададена година (1592..4902). извежда деня от седмицата. За всяка дата от указания диапазон номера на деня от седмицата (0 – за неделя, 6 – за събота) е равен на остатъка при целочислено деление на 7 на израза [2,6.m-0,2]+d+y+[y/4]+[c/4]-2.c, където d – номер на деня в месеца; m – номер на месеца в годината (1 – март, януари и февруари – 11 и 12, но от*

предходната година); c – числото образувано от първите две цифри на годината; y – числото, образувано от последните две цифри на годината.

```
#include<iostream>
#include<stdlib.h>
using namespace std;
int main()
{ int d,m,g,r,t,x,y;
 system("chcp 1251");
 cout<<"Въведи дата:";cin>>d;
 cout<<"Въведи месец:";cin>>m;
 cout<<"Въведи година:";cin>>g;
 if (m>2){t=2.6*(m-2)-0.2;x=g/100,y=g%100;} //3
 else  {t=2.6*(m+10)-0.2; x=(g-1)/100,y=(g-1)%100;}
 r=t+d+y+y/4+x/4-2*x;
 switch (r%7)
  {case 0: cout<<"неделя"<<endl; break;
   case 1: cout<<"понеделник"<<endl; break;
   case 2: cout<<"вторник"<<endl; break;
   case 3: cout<<"сряда"<<endl; break;
   case 4: cout<<"четвъртък"<<endl; break;
   case 5: cout<<"петък"<<endl; break;
   case 6: cout<<"събота"<<endl;
  }
 return 0;
}
```

Тук се дава наготово модела на задачата, за да се концентрира вниманието върху алгоритмичните и езикови елементи.

Тази програма дава възможност да се обърне вниманието на обучаемите върху съответствието на типовете на променливата и израза в оператора за присвояване (//3). Освен това се упражняват операциите с цели числа (целочислено деление и остатък при целочислено деление). Задачата предоставя възможност да се коментират особеностите на аритметичната операция деление. Използва се и операторът за избор на вариант.

3.2 Оператори за цикъл.

Реализацията на циклични процеси е фундаментална тема в програмирането. Генезисът на тези процеси е многократното изчисление.

Всеки цикличен процес има четири основни елемента: начална инициализация, условие за край, подготовка за следващо повторение, тяло на цикъла.

Един от трудните моменти при съставяне на циклични програми – това е началната инициализация на параметрите на цикъла (//4). Тези стойности изискват внимателен подбор, защото неправилните начални стойности могат да доведат до грешни резултати в алгоритмично правилна програма. Освен това чрез този подбор трябва да се гарантира завършването на цикъла.

За усвояване на цикличните процеси може да се използва проста логическа игра.

✓ *" Познай числото". Това е развиваща логическа игра. Същността на играта е в това компютърът да познае числото намислено от потребителя с най-малко опити.* Оптимално познаване на числото е троично търсене.

```
#include <iostream>
#include<stdlib.h>
using namespace std;
int main()
{ int a,b,br=0,k,flag=1, ans;
  system("chcp 1251");
```

```
       srand(time(0)); int n=1+rand()%120;//генериране на число
       cout<<"Намисли си число от 1 до "<<n<<endl;
       cout<<"Ще го позная с не повече от 7 опита, като ми отговаряш само на
въпрса:"<<endl;
      cout<<"Това ли е числото. С един от трите отговора: да; нагоре ако е по-
малко; надолу, ако е по-голямо "<<endl;
      a=1;b=n;//4
      while (br<=7&&flag)
      {
       k=(a+b)/2;br++;
       cout<<"Това ли е числото: "<<k<<endl;
       cout<<"-1- надолу,0- да, 1- нагоре: ";
       cin>>ans;
       if (0==ans) flag=0;
       else if (1==ans) a=k; else b=k;
      }
      if (flag==0) cout<<"Познах от "<<br <<" опита"<<endl;
      else cout<<"Нещо шмекеруваш. Не си отговорил коректно на някой въпрос
"<<endl;
     }
```

Пропускането на инициализация на променлива може да доведе до невярна работа на програмата (неопределена стойност на резултата). За да се избегне такава грешка, трябва явно да се инициализират всички използвани променливи.

В тази тема е подходящо да се реализира (може и като самостоятелна работа) вариант на задачата "Календар", в който вместо да се въвежда дата се въвеждат само месец и година и се извежда календар за целия месец от годината.

3.3 Съставни типове данни. Масив. Низове.

✓ *"Крави и бикове". Това е също логическа игра. Тя е по-сложна от предходната и изисква, както всички логически игри, не само скорост на мисленето, но и дълбочина и точност, умения да се анализират варианти. Програмата генерира четирицифрено число (без повтарящи се цифри). Играчът трябва да прави на всяка стъпка предположение за генерираното число. Програмата му казва колко са познатите цифри (крави) и колко от тях са на точната позиция (бикове). Поставената задача трябва да се реши с минимален брой опити.*

След запознаване със структурата "масив", може да се обърне внимание върху основни операции с масиви. Разработването на тази игра може да се реализира в следните стъпки:

a) Търсене на елемент в масив (**//4**).

b) Използвайки горната конструкция, се организира цикъл за генериране на четирицифрено число без повтарящи се цифри.

c) Цикъл за проверка има ли еднакви елементи в два масива.

d) Накрая този цикъл се поставя в цикъл отброяващ опитите за познаване на числото.

```
#include<iostream>
#include<string.h>
#include<stdlib.h>
using namespace std;
int main()
{char a[4],b[4]; int i,j,n; bool s;
 system("chcp 1251");
 cout<<"Намислих си четирицифрено число без повтаряши се цифри"<<endl;
 cout<<"Опитай да го познаеш от седем пъти!"<<endl;
```

```
//цикъл за генериране на четирицифрени числа без повтарящи се цифри
do
{
  srand(time(0));n=1234+rand()%8644;//генериране на число
  s=true;
  i=3;//проверка има ли повтарящи се цифри
  while(i>=0 && s)
  {
   a[i]=char(n % 10+'0');
   j=3;
   while (a[j]!=a[i])//4
        j--;
   s=(i==j);
   n=n / 10;
   i--;
   }
 }
while (!s);
int q=0,//опити
    cow,//крави
    bull;//бикове
do
{
 cout<<"въведи число:"<<endl;
  q++;cow=0,bull=0;
 //цикъл за проверка има ли еднакви елементи в два масива
  for (i=0; i<4;i++)
  {
    cin>>b[i];
    for (n=0;n<4;n++)
    if (a[n]==b[i])
    if (i==n) bull++; else cow++;
   }
  cout<<bull<<" бика и   "<<cow<<" крави   остават ти още "<<7-q << "
опита" <<endl;
 }
while(bull!=4 &&  q<7);
if (4==bull) cout<<"Браво! Позна. "<<endl;
cout<<"Не позна. Числото е: ";
for (i=0;i<4;i++)
 cout<<a[i];
}
```

3.4 Функции. Общи положения – дефиниране, обръщение, параметри.

Най-важната идея на структурното програмиране е идеята за модулност. Това означава, че цялата програма трябва да бъде разделена на модули с един вход и един изход. Освен това трябва да се разгледат принципите на проектиране („top-down”) и поетапна детайлизация.

В основата на алгоритъма за решение на задачата лежи математическият модел. Не трябва да се пести време от разработката и изучаване на свойствата на този модел. Това ще помогне за по-доброто разбиране на задачата и намиране на най-естествения път за нейното решение. Избира се внимателно алгоритъмът за решение. Използва се представяне на данните, съответстващо на задачата.

✓ *"Бесенка"[2]. Задава се дума само с броя на буквите. Целта е да се разпознае думата с определен брой опити. Това е пример за обучаващи и трениращи игри.*

При разглеждане на темите за символни масиви и функции е удачно да се реши тази задача на следните стъпки:

a) Функция за проверка има ли даден символ в низ;

b) Проверка въвеждана ли е буквата с използване на горната функция;

c) Проверка има ли я въведената буква и на каква позиция с използване на горната функция;

d) Цикъл за броене на опити.

```
#include<iostream>
#include<string.h>
#include<stdlib.h>
using namespace std;
const    int    n=14;    string    dumi[n]={"компютър","монитор","мишка",
"клавиатура","процесор","флашка","принтер","информатика","програмиране",
"програма","интернет","оператор","функция","променлива"};
//Функция за проверка има ли даден символ в низ
bool imali(int k,char a,char d[])
{
 for (int i=0;i<k;i++)
 if (d[i]==a)return true;
 return false;
 }
int main()
{int   i,j,k,l;bool b,b1;
 char   ch,  bukvi[30]={'0'}; string s;
 system("chcp 1251");
 srand(time(0));k=rand()%n;
 s=dumi[k];
 k=s.size();
 for (i=0;i<k;i++)
  cout<<"_ ";
 cout<<endl;
 j=5;//опити
 l=k;  int t=0;
 //цикъл за броене на опити
 do
  {b1=false;
   b=false;
   cout<<"Въведи буква на кирилица:";
   cin>>ch;
  //проверка въвеждана ли е буквата
   if (imali(t,ch, bukvi)) { j--;b=true; }
   else {bukvi[t]=ch;t++;}
   // проверка има ли я въведената буква и на каква позиция
    for (i=0;i<k;i++)
     { if (ch==s[i]&& !b) {l--;b1=true;}
       if (imali(t,s[i],bukvi))cout<<s[i]<<' ';
       else cout<<"_ ";
      }
   cout<<endl;
   if (!b1) j--;
   }
 while(l>0 && j>0);
 if (0==l) cout<<"Браво! Позна!"<<endl;
 else {
       cout<<"Съжалявам! Обесен си!"<<endl;
       cout<<" Думата е:"<<s<<endl;
```

```
        }
 return 0;
}
```

4 Заключение.

Разбира се тук не става дума за абсолютизиране на такъв подход, а само като допълнително поддържане на интереса на обучаемите към програмирането. В много от тези игрови задачи могат сериозно да се обогатят и математическите знания и умения на обучаемите [3].

Проблемите в уводния курс по програмиране се дължат основно на фактори като липса на знания за предметната област и неспособност за абстрактно мислене. И все пак когато има мотивация, интерес и проактивност по отношение на някакво съдържание част от бариерите могат да се преодолеят.

Идеята, която се предлага е особено подходяща за началното обучение по програмиране. Може да се използва интереса на обучаемите към компютърните игри като по този начин се виждат практически приложения на програмирането.

Разработването на компютърни игри по време на обучението може да мотивира обучаемите, да събуди интерес към нови теми, да облекчи обучението и да стимулира запомнянето на вече изученото.

Игрите могат да се задават като курсова работа(проект) и тогава обучаемите работят самостоятелно, решавайки сами поставените проблеми. Това повдига самочувствието им и ги подготвя за реалната дейност. Тези задачи се решават след усвояване на необходимия инструментариум.

Игрите не трябва да създават заблудата у обучаемите, че ученето е игра. Затова е добре да им се покаже и другата страна, не е лесно да се създаде интересна игра. В обучението по програмиране има добра възможност за това.

ЛИТЕРАТУРА:

[1] Гнездилова Г. А. и др., Персональный компьютер в играх и задачах,Москва,(1988).

[2] Клейман Г.М., Школы будущего: компьтеры в процессе обучения, "Радио и связь",Москва, (1987).

[3] Теодосиева, Г., Теодосиев Т., Игровите задачи в началното обучение по програмиране, Сборник доклади на 31-вата Пролетна конференция на СМБ, София, (2002), 285-289 http://www.math.bas.bg/smb/2001_2003_PK/2002/pdf/285-289.pdf

[4] Piteira, M., Costa, C., Learning computer programming. In Proceedings of the 2013 International Conference on Information Systems and Design of Communication - ISDOC '13, Lisboa, Portugal: ACM Press, (2013), 5–80, doi:10.1145/2503859.2503871.

[5] Vahldick, A., Mendes, A. J., Marcelino, M. J. A review of games designed to improve introductory computer programming competencies, 2014 IEEE Frontiers in Education Conference (FIE) Proceedings, Madrid, (2014), 1-7.

Теодоси Кирилов Теодосиев
Шуменски университет „ Епископ К.Преславски“, ФМИ
E-mail: t.teodosiev@shu.bg

Галина Йорданова Теодосиева
ППМГ „Нанчо Попович”
E-mail: pmg.g.teodosieva@abv.bg