Актуальні питання навчання

УДК 378.147

М.А. Бондаренко, В.А. Жилін, Д.П. Панасенко

Українська інженерно-педагогічна академія, Харків

РОЗРОБКА ПІДСИСТЕМИ КОНТРОЛЮ ЗНАНЬ ЗА ТЕМОЮ «ПРОГРАМУВАННЯ НА МОВІ DELPHI. ЗАПИС АРИФМЕТИЧНИХ ВИРАЗІВ»

Дана робота присвячена організації підсистеми контролю знань учнів за темою «Програмування на мові Delphi. Запис арифметичних виразів». У роботі представлені результати розробки програмного й інформаційного забезпечення такої підсистеми. Наведені результати тестування підсистеми контролю знань, які показують, що розроблений програмний продукт працює відповідно до технічного завдання. Дана програма може бути використана в навчальному процесі як незалежно, так і у складі глобальної системи контролю знань.

***Ключові слова**: автоматизований контроль знань, об'єктно-орієнтоване програмування, програмне й інформаційне забезпечення підсистем контролю знань.*

Вступ

Аналіз сучасного стану досліджень за напрямком. Останніми роками методи автоматизованого контролю знань набувають все більшого поширення. Зазвичай, вони реалізують певні тести, в яких пропонується набір питань і декілька варіантів відповідей на кожне питання. При цьому серед варіантів відповідей є тільки один правильний. Якщо учень вибирає відповіді випадковим чином, то при достатній кількості питань вірогідність отримання позитивної оцінки надзвичайно мала. У роботі [1] розглядається зміст дисципліни «Інформаційні технології і програмування в середовищі Delphi», де основним є об'єктно-орієнтоване програмування. Програмування виникло і розвивалося як процедурне програмування. Процедурний підхід допускає, що основою програми є алгоритм, тобто деяка процедура обробки даних. Ускладнення завдань, що вирішуються на ЕОМ, збільшення їх масштабів, проявили недоліки процедурного підходу, пов'язані з надійністю програм, повторним використанням коду тощо. В результаті багаторічного досвіду й аналізу цих проблем і методів на початку 90-х років минулого століття з'явилася концепція об'єктного орієнтованого програмування (ООП). Програми, розроблені старими методами, без використання об'єктного орієнтованого підходу, не є «неправильними». У більшості випадків вони чудово працюють і нічого в них змінювати не треба. А ось при розробці зовсім нових систем програмування використання ООП може дати безліч переваг [2, 3]. Тому, незважаючи на те, що за останні 10 років об'єктні орієнтовані технології розвивалися досить швидко і стабільно, особливий поштовх їхнього розвитку дала поява Інтернету. Він відкрив перед ООП область зовсім нових розробок, без необхідності використання старих методів і засобів. Сьогодні більшість баз даних і Web-технологій є за своєю природою об'єктно-орієнтованими.

Постановка задачі. Метод контролю знань, що розглядається в цій роботі, не може претендувати на об'єктивну і всеосяжну оцінку знань учнів. Він ніколи не замінить такі форми контролю, за яких виявляється творче сприйняття знань, здатність учнів самостійно мислити і формулювати відповіді на поставлені питання, уміння застосовувати отримані знання з практики (семінари, обговорення, практичні заняття, лабораторні роботи, заліки, іспити, тощо). В той же час, тестування цілком може застосовуватися в навчальному процесі для оперативного контролю учнів, первинної, «грубої», оцінки засвоєння отриманих знань і навіть як непрямий спосіб закріплення пройденого матеріалу. Таким чином, метод контролю знань за допомогою тестування є актуальним. Немає сумнівів, що й надалі він набуватиме все більшого поширення (спільно з іншими формами контролю).

Необхідно відмітити, що створення систем тестування природним чином розпадається на дві сновні задачі: 1) розробка відповідного програмного забезпечення; 2) розробка набору питань і варіантів відповідей до них (інформаційного забезпечення).

І, якщо перша задача може бути з часом в тому або іншому ступені уніфікована, то друга вимагає творчого підходу з боку викладача. Саме від нього залежить, чи буде тест містити достатню кількість питань (щоб унеможливити випадкове отримання позитивної оцінки); наскільки повно набір питань охоплює пройдений матеріал; наскільки «правдоподібно» сформульовані варіанти відповідей (щоб правильну відповідь не можна було вибрати простим виключенням явно непідходящих варіантів відповіді). Дана робота присвячена рішенню обох за-

значених задач, а саме – розробці програмного та інформаційного забезпечення підсистеми контролю знань за темою «Програмування на мові Delphi. Запис арифметичних виразів».

Виклад основного матеріалу

Як вже відзначалося, підсистема контролю знань складається з двох складових – програмного та інформаційного забезпечення. До кожної з цих складових мають бути висунуті певні вимоги.

Передусім, програмне забезпечення підсистеми контролю знань повинне виключити можливість, коли учень може просто «механічно» запам'ятати, які номери варіантів відповідей є правильними. Наприклад, після декількох проходжень одного і того ж тесту і обговорення результатів з викладачем учень запам'ятовує, що на питання №1 треба відповідати варіантом №3, на питання №2 – варіантом №4 і т.д. Щоб виключити таку можливість, підсистема контролю знань при кожному новому запуску тесту повинна пропонувати варіанти відповідей до питань у випадковому порядку. Ще краще буде, якщо і питання кожного разу з'являтимуться у випадковому порядку. В цьому випадку учневі доведеться запам'ятовувати і зміст питання, і зміст правильної відповіді до нього, що вже саме по собі сприяє змістовному засвоєнню матеріалу.

Крім того, програмне забезпечення повинне давати учневі можливість повертатися до вже пройдених питань і, за необхідності, вибирати інші варіанти, ніж ті, які він зазначив раніше. Цим виключається вірогідність випадкового вибору помилкових варіантів.

Ще однією вимогою до програмного забезпечення підсистеми контролю знань є перевірка того, що учень відповів повністю на усі питання тесту. Інакше можливе випадкове отримання позитивної оцінки (наприклад, дано правильні відповіді на 8 питань з 10, а на 2 питання відповідей взагалі не було), тоді як відсутність відповідей на яке-небудь питання свідчить про наявність в учня пропусків в знаннях із заданою теми.

Дуже важливою вимогою до програмного забезпечення є можливість викладача після закінчення тестування проглянути, які саме варіанти відповідей обирав учень. По-перше, це дозволяє виявити типові помилки, які, швидше за все, пов'язані із складністю матеріалу або з недоліками у викладанні. По-друге, результати тестування корисно обговорити з учнями, щоб допомогти їм розібратися у своїх помилках і, нарешті, краще засвоїти матеріал.

Основною вимогою до інформаційного забезпечення є наявність достатньої кількості питань із заданою темою. Це необхідно для того, щоб понизити вірогідність випадкового отримання позитивної оцінки і підвищити точність та об'єктивність результатів тестування. Можна стверджувати, що мінімальна кількість питань в тесті має бути в два рази більша, ніж максимальний бал, за яким оцінюються знання (наприклад, при п'ятибальній системі в тесті має бути як мінімум десять питань, при 12-бальній системі – 24 питання тощо).

З цієї ж причини може бути висунута вимога до кількості варіантів відповідей. На нашу думку, оптимальна кількість варіантів відповідей на кожне питання складає від 4 до 6 варіантів. Менша кількість варіантів підвищує вірогідність випадкового вгадування правильної відповіді, більша – ускладнює учневі правильне осмислення змісту варіантів відповідей.

Обов'язковою вимогою до помилкових варіантів відповідей є їх «правдоподібність». Відповіді мають бути сформульовані так, щоб учень не міг суто логічними висновками відкинути явно непідходящі варіанти. Вони мають бути «схожими» на правильну відповідь.

Цілком зрозумілою вимогою є повнота тесту, тобто питання повинні по можливості охоплювати увесь матеріал із заданої теми.

Таким чином, в результаті проведеного аналізу можна сформулювати наступні загальні вимоги до підсистеми контролю знань, що необхідно враховувати при її розробці.

Вимоги до програмного забезпечення:

1) програма має забезпечувати появу варіантів відповіді на кожне питання у випадковому порядку (як мінімум). У найкращому випадку і самі питання, і варіанти відповідей до них повинні з'являтися у випадковому порядку;

2) програма має давати учневі можливість повертатися до вже пройдених питань і при необхідності змінювати вибір варіанту відповіді;

3) програма має забезпечувати вимогу відповіді на усі питання тесту;

4) програма має давати викладачеві можливість після закінчення тестування проглянути, які варіанти відповідей вибирав учень.

Вимоги до інформаційного забезпечення:

1) тест має містити достатню кількість питань (як мінімум в два рази більше, ніж максимальний бал);

2) питання тесту повинні якнайповніше охоплювати усю тему;

3) для кожного питання має бути не менше чотирьох варіантів відповіді;

4) варіанти відповідей повинні виглядати «правдоподібними».

Розглянемо питання вибору параметрів програмного та інформаційного забезпечення підсистеми контролю знань.

Проведений аналіз показав, що можливі, як мінімум, два підходи до організації даних. При першому підході дані тесту (питання, варіанти відповідей, інформація про правильні відповіді) зберігаються в початковому коді програми. Основна перевага тут полягає в тому, що ці дані повністю захи-

щені від зовнішнього «вивчення», в результаті якого можна було б отримати інформацію про правильні відповіді. Учень працює тільки з одним виконуваним файлом (*.exe) без будь-яких додаткових даних. Головний недолік такого підходу полягає в тому, що для зміни наявного тесту або розробки нового необхідно мати початковий текст програми і вміти вносити до нього необхідні зміни.

Другий підхід передбачає зберігання даних тесту в зовнішніх файлах. Основна перевага такого підходу полягає в універсальності програмного забезпечення. Програма, що один раз відкомпілювалася, може використовуватися як основа для будь-якого тесту. Розроблюється і змінюється тільки інформаційне забезпечення, яке може готуватися за допомогою звичайних текстових редакторів. Проте в цій універсальності полягає і основний недолік. Якщо зовнішні дані залишити в початковому виді, то вони можуть бути легко прочитані, тобто учнями може бути отримана інформація про правильні відповіді. Щоб цього уникнути, дані необхідно «шифрувати» (кодувати), а це викликає необхідність додаткового програмного забезпечення.

В даній роботі вибрано такі параметри програмного й інформаційного забезпечення:

- дані тесту зберігаються в окремому файлі в закодованому виді;
- тест розрахований на 10 питань;
- до кожного питання пропонується 4 варіанти відповіді, тільки один з яких є правильним;
- програма розрахована на оцінку за 5-бальною системою;
- програма пред'являє і питання, і варіанти відповідей у випадковому порядку;
- програма допускає переходи від одного питання до іншого у будь-якому порядку з можливістю змін вибору варіанту відповіді.

В результаті проведеного аналізу визначено основні вимоги до програмного та інформаційного-забезпечення і вибрано їх параметри, що дає можливість безпосередньо приступити до розробки підсистеми контролю знань.

Діалогове вікно підсистеми контролю знань реалізовано наступною формою, розробленою в середовищі Delphi (рис. 1).

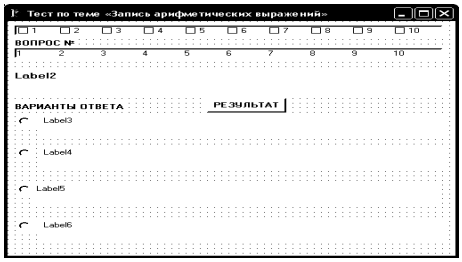

Рис. 1. Діалогове вікно підсистеми контролю знань

Мітка Label2 використовується для відображення поточного питання. Мітки Label3 – Label6 містять чотири варіанти відповіді на поточне питання. Мітка Label1 відображає номер поточного питання, мітка Label7 – напис «Ваша відповідь».

Елементи управління RadioButton1 – RadioButton4 дозволяють користувачеві вказати вибраний варіант відповіді шляхом натиснення лівою клавішею миші при наведенні покажчика на білий кружок біля варіанту відповіді.

У верхній частині вікна (над написом «Питання №») знаходиться елемент управління CheckListBox, який містить список з 10 чисел (номерів питань). В процесі роботи у ньому відображатиметься, на які питання була дана відповідь, а на які відповідей не було.

Під написом «Питання №» знаходиться елемент управління ListBox, який також містить список з 10 чисел (номерів питань). За допомогою цього списку можна перейти до будь-якого питання тесту в довільному порядку.

Кнопка «Результат» дозволяє отримати результати тестування.

При запуску програми ця кнопка невидима і стає доступною тільки після того, як учень відповів на усі питання.

При розробці програми виберемо наступну структуру даних. Передусім визначимо типи даних:

1) тип Otv, що є записом з полями: Otvet – рядок, що містить один варіант відповіді; Prav – логічна ознака, що показує, чи є цей варіант відповіді правильним; Uch – логічна ознака, що показує, чи вибрав учень цей варіант відповіді у якості правильної;

```
type
Otv = record
Otvet: string;
Prav: boolean;
Uch: boolean
end;
```

2) тип VVO (питання і варіанти відповіді) – запис, що містить питання і чотири варіанти відповіді до нього.

```
VVO = record
Vopros: string;
VarOtv: array[1.4] of Otv
end;
```

3) тип, що описує цілочисельний масив з чотирьох чисел:

```
AR4 = array [1.4] of integer;
```

4) тип, що описує цілочисельний масив з десяти чисел:

```
AR10 = array [1.10] of integer.
```

Типи AR4 і AR10 потрібні для отримання випадкових послідовностей з 4 і 10 чисел.

Крім того, оголосимо наступні глобальні змінні:

– масив M, що містить 10 питань тесту і варіанти відповідей до них: M: array [1 . 10] of VVO;

– цілочисельна змінна NomVopr, яка в процесі роботи міститиме номер поточного питання:

NomVopr: integer;

– змінні SO (список відповідей) і SV (список питань), необхідні для зберігання послідовності випадкових чисел : SO: AR4; SV: AR10;

Розробимо наступні процедури і функції.

– ***Функція Decoding***

Ця функція в якості вхідного аргументу приймає рядок, що містить текст в закодованому виді. Результатом функції є цей же текст в розкодованому виді.

Нижче приведений повний текст цієї функції.

```
function Decoding(S: String) : String;
var k, L: integer; SC: string;
C: Char; X: byte;
Begin L:=Length(S); SC:=S;
for k:=1 to L do
begin C:=S[k]; X:=287 - Ord(C); SC[k]:=Chr(X);
end;
Result:=SC;
end;
```

– ***Процедура Rand4*** (var RA: AR4).

Ця процедура не має вхідних аргументів. В якості результату процедура повертає цілочисельний масив з чотирьох чисел, що містить різні випадкові числа від 1 до 4. В якості джерела випадкових чисел використані стандартні функції Randomize і Random.

Текст цієї процедури має вигляд:

```
procedure Rand4(var RA: AR4);
var i, k, n: integer;
begin for i:=1 to 4 do RA[i]:=0; n:=0;
randomize; while n < 4 do
begin k:=Random+1; for i:=1 to n do
begin if RA[i] = k then begin k:=0;
beak
end; end;
if k <> 0 then
begin n:= n+1; RA[n]:= k;
end; end; end;
```

Процедура Rand4 працює таким чином. Спочатку за допомогою стандартної функції Random отримується випадкове ціле число від 1 до 4. Далі перевіряється, чи не міститься вже таке число у вихідному масиві RA. Якщо такого числа ще немає, воно записується в перший вільний елемент цього масиву. Якщо ж таке число вже є в масиві RA, виконується повернення до отримання випадкового числа і перевірки його входження в масив RA. Після закінчення роботи масив RA міститиме різні числа від 1 до 4 у випадковому порядку. Наприклад, RA[1] дорівнюватиме 2, RA[2] – 4, RA[3] – 1, RA[4] – 3.

Процедура Rand4 використовується для відображення варіантів відповіді на питання у випадковому порядку. В якості першого варіанту відображатиметься відповідь, номер якої міститься в першому елементі масиву RA, другим варіантом буде відповідь, номер якої міститься в другому елементі масиву RA і т.д. Тут під номером варіанту розуміється його положення в масиві Vopros відносно самого питання. Тобто варіант відповіді, що йде першим після питання в масиві Vopros має перший номер, наступний варіант - другий і т. д.

– ***Процедура Rand10*** (var RA: AR4).

Ця процедура не має вхідних аргументів. В якості результату процедура повертає цілочисельний масив з десяти чисел, що містить різні випадкові числа від 1 до 10. В якості джерела випадкових чисел використані стандартні функції Randomize і Random.

Нижче приведений текст цієї процедури.

```
procedure Rand10(var RA: AR10);
var i, k, n: integer;
begin for i:=1 to 10 do RA[i]:=0; n:=0;
Randomize;
while n < 10 do
begin k:=Random+1; for i:=1 to n do
begin if RA[i]=k then
begin k:=0;
Break
end; end;
if k<>0 then
begin n:=n+1; RA[n]:=k;
end; end; end;
```

Робота цієї процедури аналогічно процедурі Rand4.

– ***Процедура PokazVoprosa***.

Ця процедура не має вхідних і вихідних параметрів. Вона використовується для відображення в діалоговому вікні питання і варіантів відповіді до нього. Номер питання має бути заданий в глобальній змінній NomVopr. Нижче приведений текст цієї процедури.

```
procedure PokazVopros;
var i, k: integer;
begin i:=SV[NomVopr];
Form1.Label2.Caption:=M[i].Vopros;
Rand4(SO); k:=SO[1];
Form1.Label3.Caption:=M[i].VarOtv[k].Otvet;
Form1.RadioButton1.Checked:=M[i].VarOtv[k].Uch;
k:=SO[2];
Form1.Label4.Caption:=M[i].VarOtv[k].Otvet;
Form1.RadioButton2.Checked:=M[i].VarOtv[k].Uch;
k:=SO[3];
Form1.Label5.Caption:=M[i].VarOtv[k].Otvet;
Form1.RadioButton3.Checked:=M[i].VarOtv[k].Uch;
k:=SO[4];
Form1.Label6.Caption:=M[i].VarOtv[k].Otvet;
Form1.RadioButton4.Checked:=M[i].VarOtv[k].Uch;
Form1.ListBox1.ItemIndex:=NomVopr - 1;
Form1.CheckListBox1.ItemIndex:=NomVopr - 1;
end;
```

Процедура працює таким чином. Спочатку зі списку випадкових номерів питань вибирається поточний випадковий номер. Питання з цим номером

відображається у вікні. Далі, за допомогою описаної раніше процедури Rand4 визначається випадковий порядок, в якому відображаються варіанти відповіді. У цьому порядку вони відображаються у вікні. У елементи управління заноситься значення поля Uch(відповідь учня). Поточний номер питання відзначається в елементах управління ListBox і CheckListBox.

– ***Процедура Gotov***.

Ця процедура перевіряє, чи були дані відповіді на усі питання тесту. Якщо це так, то видається відповідне повідомлення і стає доступною кнопка «Результат». Нижче приведений текст цієї процедури.

```
procedure Gotov;
var i: integer; R: boolean;
begin
if Form1.Button1.Visible then exit; R:=true;
for i:=0 to 9 do R:=R and
Form1.CheckListBox1.Checked[i];
if R then
begin
ShowMessage('ВИ ВІДПОВІЛИ НА УСІ
ПИТАННЯ'+chr(13)+
'Щоб упізнати результат тестування, '+
'натисніть на кнопку «РЕЗУЛЬ-
      ТАТ»'+chr+'Перед цим '+
'Ви можете ще раз проглянути свої відповіді і, '+
'при необхідності, змінити їх.');
Form1.Button1.Visible:=true;
end; end;
```

Розглянемо програмування подій.

Подія FormCreate.

Ця подія настає відразу після появи діалогового вікна програми на дисплеї. При цьому здійснюються наступні дії: відкривається текстовий файл Test.txt, що містить питання і варіанти відповідей в закодованому виді. Файл читається по рядках і кожен рядок розкодується за допомогою функції Decoding. Розкодовані питання, варіанти відповідей і ознаки правильності заносяться в глобальний масив M. Після закриття файлу за допомогою процедури Rand10 будується список випадкових чисел від 1 до 10. У цьому порядку відображатимуться питання при цьому запуску тесту. Глобальна змінна NomVopr, що містить номер поточного питання набуває значення 1 і викликається процедура PokazVoprosa для відображення поточного(першого) питання.

Процедура обробки цієї події має вигляд:

```
procedure TForm1.FormCreate(Sender: TObject);
var i, k: integer; S: string; F: TextFile;
begin AssignFile(F, 'Test.tst'); Reset(F); S:=''; i:=1;
while(not Eof(F)) and(i<11) do
begin Readln(F, S); M[i].Vopros:=Decoding(S);
for k:=1 to 4 do
begin Readln(F, S);
M[i].VarOtv[k].Otvet:=Decoding(S);
M[i].VarOtv[k].Prav:=false;
M[i].VarOtv[k].Uch:=false;
end;
M[i].VarOtv[1].Prav:=true; i:=i+1
end;
CloseFile(F); Rand10(SV); NomVopr:=1;
PokazVoprosa;
end;
```

Події RadioButton1Click - RadioButton4Click.

Ці події настають при натисненні на один з елементів управління RadioButton при виборі варіанту відповіді. Вибрана відповідь заноситься у відповідне поле Uch масиву M. Якщо на це питання ще не була дана відповідь, то у відповідному полі елементу CheckListBox ставиться відмітка. Далі викликається процедура Gotov для перевірки закінчення питань. Якщо номер поточного питання менше 10 (загального числа питань в тісті), то програма автоматично збільшує номер питання на 1 і пропонує наступне питання.

Нижче наведені процедури обробки цих подій:

```
procedure TForm1.RadioButton1Click
(Sender: TObject);
Var i, j, k: integer;
begin i:=SV[NomVopr]; k:=SO[1];
for j:=1 to 4 do M[i].VarOtv[j].Uch:=false;
M[i].VarOtv[k].Uch:=true;
if Form1.CheckListBox1.Checked[NomVopr - 1] then
      exit;
Form1.CheckListBox1.Checked[NomVopr - 1]:=true;
Gotov; if NomVopr<10 then
begin NomVopr:=NomVopr+1; PokazVoprosa;
end; end;
procedure TForm1.RadioButton2Click
(Sender: TObject);
Var i, j, k: integer;
begin i:=SV[NomVopr]; k:=SO[2];
for j:=1 to 4 do M[i].VarOtv[j].Uch:=false;
M[i].VarOtv[k].Uch:=true;
if Form1.CheckListBox1.Checked[NomVopr - 1] then
      exit;
Form1.CheckListBox1.Checked[NomVopr - 1]:=true;
Gotov; if NomVopr<10 then
begin NomVopr:=NomVopr+1; PokazVoprosa;
end; end;
procedure TForm1.RadioButton3Click
(Sender: TObject);
Var i, j, k: integer;
begin i:=SV[NomVopr]; k:=SO[3];
for j:=1 to 4 do M[i].VarOtv[j].Uch:=false;
M[i].VarOtv[k].Uch:=true;
if Form1.CheckListBox1.Checked[NomVopr - 1] then
      exit;
Form1.CheckListBox1.Checked[NomVopr - 1]:=true;
Gotov;
if NomVopr<10 then
begin NomVopr:=NomVopr+1; PokazVoprosa;
end; end;
procedure TForm1.RadioButton4Click
(Sender: TObject);
```

```
Var i, j, k: integer;
begin i:=SV[NomVopr]; k:=SO[4];
for j:=1 to 4 do M[i].VarOtv[j].Uch:=false;
M[i].VarOtv[k].Uch:=true;
if Form1.CheckListBox1.Checked[NomVopr - 1] then
      exit;
Form1.CheckListBox1.Checked[NomVopr - 1]:=true;
Gotov;
if NomVopr<10 then
begin NomVopr:=NomVopr+1; PokazVoprosa;
end; end;
```

Подія ListBox1Click

Ця подія настає при натисненні лівою клавішею миші на який-небудь номер питання в елементі ListBox1. При цьому відбувається перехід до цього питання.

Нижче приведена процедура обробки цієї події.

```
procedure TForm1.ListBox1Click
(Sender: TObject);
egin NomVopr:=ListBox1.ItemIndex+1;
PokazVoprosa;
end;
```

Подія Button1Click

Ця подія настає при натисненні на кнопку «Результат». При цьому підраховується кількість правильних відповідей і виходячи з цього визначається оцінка за 5-бальною шкалою. Питання і вибрані варіанти відповідей записуються у файл Результат_Тесту.txt і програма завершує свою роботу.

Нижче приведена процедура обробки цієї події:

```
procedure TForm1.Button1Click
(Sender: TObject);
Var i, j, k, n, bal: integer; F: TextFile; S: string;
begin n:=0; for i:=1 to 10 do
begin for k:=1 to 4 do
if(M[i].VarOtv[k].Prav)
and(M[i].VarOtv[k].Uch) then n:=n+1;
end;
case n of
   0..2: bal:=1;
   3..4: bal:=2;
   5..6: bal:=3;
   7..8: bal:=4;
   9..10: bal:=5
end;
AssignFile(F,'Результат_Тесту.txt'); Rewrite(F);
for j:=1 to 10 do
begin i:=SV[j];
S:='Питання №'+IntToStr+'. '+M[i].Vopros;
writeln(F, S);
S:='Ваша відповідь: ';
for k:=1 to 4 do if M[i].VarOtv[k].Uch then
begin S:=S+M[i].VarOtv[k].Otvet; writeln(F, S);
if M[i].VarOtv[k].Prav then
S:='ЦЕ ПРАВИЛЬНА ВІДПОВІДЬ!'
else
S:='ЦЕ ПОМИЛКОВА ВІДПОВІДЬ!';
end;
writeln(F, S); writeln(F);
end;
S:='Правильних відповідей: '+IntToStr(n);
writeln(F, S);
S:='Ваша оцінка: '+IntToStr(bal);
writeln(F, S); CloseFile(F);
Showmessage('Правильних відповідей :
'+IntToStr(n)+chr(13)+
'Ваша оцінка: '+IntToStr+ chr(13)+
'Детальніше результат тестування Ви можете подивитися у '+     'файлі Результат_Тесту.txt');
Halt(0); end;
```

Програма має простий і зрозумілий інтерфейс. Уся робота з програмою здійснюється тільки за допомогою миші. Після запуску програми на екрані з'являється діалогове вікно програми з першим питанням тесту. Питання і варіанти відповідей приведені у випадковому порядку (рис. 2).

У верхній частині вікна розташовується список номерів питань з квадратиками. У цих квадратиках відзначається, чи була дана відповідь на питання з цим номером. Цей список недоступний користувачеві для змін. Аналогічний список номерів (але без квадратиків) показаний нижче напису «Питання №». У ньому відзначається номер поточного питання. В цьому списку можна клацнути мишкою по будь-якому номеру і відразу перейти на вказане питання.

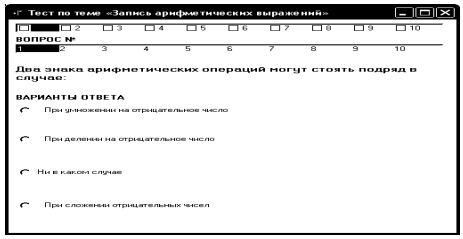

Рис. 2. Після запуску програми відображається перше питання тесту

Правильний варіант відповіді вказується шляхом натиснення лівою клавішею миші у білому кружечку, розташованому біля вибраної відповіді (рис. 3).

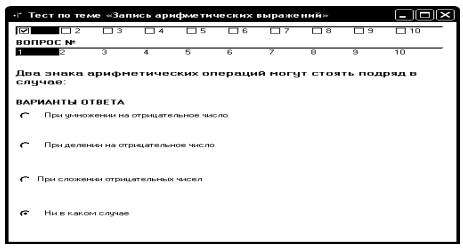

Рис. 3. Вибір варіанту відповіді

Після вибору варіанту відповіді програма автоматично перейде до наступного питання. У будь-який момент можна перейти до будь-якого питання і, за необхідності, змінити вибір варіанту відповіді. Коли усі питання пройдені, з'являється наступне повідомлення (рис. 4).

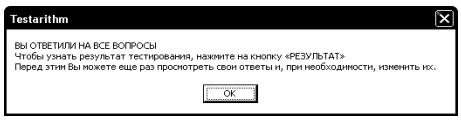

Рис. 4. Повідомлення про те, що усі питання вичерпані

У середній частині вікна з'являється кнопка «Результат». До того, як натиснути на цю кнопку для отримання результату тестування, учень може ще раз проглянути свій вибір варіантів відповідей з усіх питань і, за необхідності, змінити його.

Результат тестування може бути пред'явлений викладачеві (рис. 5). Після натиснення кнопки OK відбувається остаточний вихід з програми(вікно закривається).

Детальніше результати тестування можна проглянути в текстовому файлі Результат_Тесту.txt, який створюється в тій же теці, з якої був проведений запуск програми. У ньому за допомогою будь-якого текстового редактора (наприклад, БлокнотаWindows) можна проглянути усі питання і вибрані варіанти відповідей, а також побачити, які з них були правильними, а які – помилковими (рис. 6).

Рис. 5. Результати тестування

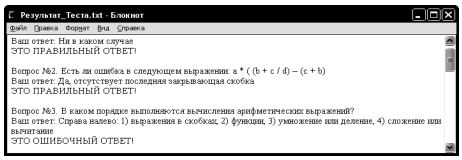

Рис. 6. Перегляд результатів тестування в текстовому файлі(фрагмент)

Таким чином, розроблена і протестована підсистема контролю знань за темою «Програмування на мові Delphi. Запис арифметичних виразів». Підсистема задовольняє основним вимогам, що пред'являються до програмного та інформаційного забезпечення для таких систем.

В результаті виконаної роботи був проведений аналіз загальних вимог, що висуваються до підсистем контролю знань, розроблено відповідне програмне та інформаційне забезпечення для підсистеми контролю знань за темою «Програмування на мові Delphi. Запис арифметичних виразів». Показані приклади роботи програми, що демонструють коректність її функціонування. Розроблена підсистема може бути використана для оперативного контролю знань як автономно, так і у складі структурно більш об'ємної системи контролю знань.

Перспективи подальших досліджень

Подальші дослідження мають бути спрямованими на постійне вивчення проблем розробки підсистем контролю знань з дисциплін, що стосуються прикладного об'єктно-орієнтованого програмування і призначені для інженерно-педагогічних спеціальностей. Такі дослідження мають передбачати використання інноваційних методів навчання, а також розробку новітніх систем контролю знань при підготовці майбутніх висококваліфікованих фахівців.

Список літератури

1. Бондаренко М.А. Інформаційні технології та програмування у середовищі Delphi, гриф МОНУ / М.А. Бондаренко. – Х.: ФОП Александрова К.М., 2014. – 536 с.

2. Бондаренко М.А. Програмування на Object Pascal в середовищі Delphi 6 / М.А. Бондаренко. – Х.: «Бізнес Інформ», 2003. – 704 с.

3. Бобровский С. Delphi 5: Учебный курс / С. Бобровский. – СПб.: Питер, 2002. – 640 с.

Надійшла до редколегії 25.04.2014

Рецензент: д-р техн. наук, проф. І.В. Рубан, Харківський університет Повітряних Сил ім. І.Кожедуба, Харків.

РАЗРАБОТКА ПОДСИСТЕМЫ КОНТРОЛЯ ЗНАНИЙ ПО ТЕМЕ «ПРОГРАММИРОВАНИЕ НА ЯЗЫКЕ DELPHI. ЗАПИСЬ АРИФМЕТИЧЕСКИХ ВЫРАЖЕНИЙ»

Н.А. Бондаренко, В.А. Жилин, Д.П. Панасенко

Данная работа посвящена организации подсистемы контроля знаний учащихся по теме «Программирование на языке Delphi. Запись арифметических выражений». В работе представлены результаты разработки программного и информационного обеспечения такой подсистемы. Приведены результаты тестирования подсистемы контроля знаний, которые показывают, что разработанный программный продукт работает в соответствии с техническим заданием. Данная программа может быть использована в учебном процессе как независимо, так и в составе глобальной системы контроля знаний.

Ключевые слова: *автоматизированный контроль знаний, объектно-ориентированное программирование, программное и информационное обеспечение подсистем контроля знаний.*

DEVELOPMENT OF ПОДСИСТЕМИ CONTROL OF KNOWLEDGES ON THE TOPIC «PROGRAMMING IN LANGUAGE OF DELPHI. RECORD OF ARITHMETIC EXPRESSIONS»

N.A. Bondarenko, V.A. Zhilin, D.P. Panasenko

This work is devoted to the organization of the subsystem control students' knowledge on the topic "Programming in Delphi. Writing arithmetic expressions. " The paper presents the results of development of software and information support of this subsystem. Results of testing knowledge-based control, which show that the developed software product operates in accordance with the technical specifications. The developed software can be used in the educational process both independently and as part of a global monitoring system of knowledge.

Keywords: *automated control of knowledge, object-oriented programming, software and information support knowledge control subsystems.*