В якості індивідуальних проектів студенти розробляють алгоритми розв'язування конкретних прикладних задач безумовної та умовної нелінійної оптимізації і описують їх однією з процедурних мов програмування у програмних середовищах за власним вибором. Результати порівнюють з наведеними вище.

Розглянуті у статті методичні аспекти навчання основ теорії і методів оптимізації з комп'ютерною підтримкою сприяють розвитку у студентів математичних та інформатичних компетентностей.

Список використаних джерел

1. Жалдак М.І., Триус Ю.В. Основи теорії і методів оптимізації: Навчальний посібник. – Черкаси: Брама-Україна, 2005. – 608 с.
2. Васильев Ф.П. Численные методы решения экстремальных задач. – М. : Наука. Главная редакция физико-математической литературы, 1980. – 518 с.
3. Кузьміна Н.М. Зміст і методика навчання курсу «Основи теорії і методів оптимізації» в педагогічному університеті // Науковий часопис НПУ імені М.П. Драгоманова. Серія № 2 Комп'ютерно-орієнтовані системи навчання: Зб. наук. праць / Редрада. – К. НПУ імені М.П. Драгоманова, 2012. – №13(20). – С. 85-89.
4. Кузьміна Н.М. Основи теорії і методів оптимізації: програма навчальної дисципліни для підготовки студентів спеціальностей 7.04030201, 8.04030201 «Інформатика*» Інституту інформатики НПУ імені М.П. Драгоманова// К.: Вид-во НПУ імені М.П. Драгоманова, 2014. – 28 с.

Рафальська М. В.

Національний педагогічний університет імені М.П. Драгоманова

Формування компетентностей у галузі алгоритмізації та програмування майбутніх вчителів інформатики у процесі навчання операційних систем

Однією з актуальних проблем у галузі підготовки майбутніх вчителів інформатики є формування у них системи предметних (інформатичних) компетентностей на рівні, достатньому для здійснення успішної професійної діяльності та подальшого самонавчання та саморозвитку [1, 44; 2, 11; 3, 36].

Аналіз досліджень, присвячених цій проблемі (зокрема, робіт В. Ю. Бикова, М. І. Жалдака, М. П. Лапчика, Н. В. Морзе, С. А. Ракова, Ю. С. Рамського, С.О. Семерікова, О. М. Спіріна, Ю. В. Триуса, С. М. Яшанова та ін.), дає змогу зробити висновок про те, що формування *системи інформатичних компетентностей майбутніх вчителів інформатики* полягає в опануванні на достатньо високому рівні змістом фундаментальних та прикладних розділів інформатики, її основними методами з врахуванням майбутньої професійної діяльності, набутті досвіду розв'язування задач професійного спрямування, опануванні методологією здійснення дослідницької діяльності у відповідній предметній галузі. До предметних (інформатичних) компетентностей майбутніх вчителів інформатики можна віднести: інформологічно-методологічні, інформаційно-технологічні компетентності, компетентності у галузі моделювання, комп'ютерної інженерії, алгоритмізації та програмування [1, 46].

В результаті оцінювання рівня сформованості інформатичних компетентностей студентів педагогічних університетів, бесід з викладачами інформатичних дисциплін було з'ясовано, що велика частка студентів мають низький рівень сформованості компетентностей у галузі алгоритмізації та програмування. Показники змінюються в залежності від курсу навчання, при цьому найнижчі показники рівня сформованості зазначених компетентностей спостерігаються на першому курсі навчання в університеті (65 % студентів мають низький рівень компетентностей у галузі алгоритмізації та програмування).

Відсутність обов'язкової державної підсумкової атестації з інформатики для випускників середніх загальноосвітніх закладів, особливості навчання інформатики у профільній школі спричинюють те, що на спеціальність «Інформатика*» вступають абітурієнти з різним рівнем інформатичних компетентностей, зокрема компетентностей у галузі алгоритмізації та програмування. Багато студентів через відсутність базових знань з основ алгоритмізації та програмування не можуть пристосуватися до темпу навчання програмування в університеті та демонструють низькі показники успішності, у той час як інші студенти готові до вивчення мов програмування та розв'язування запропонованих завдань. Через значну відмінність у рівнях сформованості компетентностей у галузі алгоритмізації та програмуванні студенти, які мають труднощі з програмуванням, втрачають мотивацію до навчання, набуття ними відповідних компетентностей гальмується.

Виявлене протиріччя між вимогами до рівня підготовки вчителів інформатики у галузі алгоритмізації та програмування та наявним рівнем сформованості відповідних компетентностей у студентів педагогічних університетів спричинює необхідність перегляду процесу формування

компетентностей у галузі алгоритмізації та програмування, зокрема підвищення рівня їх сформованості у студентів першого курсу.

Як відомо [4, 139], процес формування компетентностей має акумулюючий характер, тобто нові компетентності, яких набувають студенти у процесі навчання, інтегруються з вже сформованими, доповнюючи, поглиблюючи та розширюючи їх. Тому, у процесі навчання дисциплін інформатичного циклу, що входять до програми підготовки вчителів інформатики, доцільно добирати зміст, методи, засоби, форми організації навчання, використання яких сприяє набуттю студентами компетентностей у галузі алгоритмізації та програмування.

У статті розглядаються шляхи формування зазначених вище компетентностей у процесі навчання операційних систем Windows та Linux студентів першого курсу.

З метою усунення зазначеного протиріччя до змісту навчання операційних систем Windows та Linux доцільно включити питання, присвячені технології написання сценаріїв (скриптів) – командних файлів, використання яких дає змогу автоматизувати виконання рутинних завдань щодо налаштування параметрів середовища операційної системи, здійснення операцій з файлами, управління пристроями введення/виведення тощо. Метою навчання теми, з одного боку, є формування знань та умінь студентів щодо управління інформаційною системою на рівні вищому, ніж звичайний користувач (що входить до системи інформаційно-комунікаційних компетентностей майбутніх вчителів інформатики), а з другого – формування компетентностей щодо використання базових структур алгоритмів у процесі написання сценаріїв для розв’язування задач професійного характеру, реалізації алгоритмів з різними типами даних, а також введення, налагодження та тестування програм на комп’ютері у середовищі різних операційних систем (тобто компонентів компетентностей студентів у галузі алгоритмізації та програмування).

Розглянемо деякі методичні аспекти навчання технології написання сценаріїв засобами Windows PowerShell, а також командного інтерпретатора bash операційної системи Linux.

Windows PowerShell – оболонка операційної системи Windows з інтерфейсом командного рядка і відповідною мовою написання сценаріїв, розроблена на базі технології .NET Framework [5]. Версія 2.0 оболонки є компонентом операційних систем Windows 7, Windows 8, Windows Server 2008 R2 та використовується для управління інформаційною системою як локально, так і віддалено. До складу оболонки включено широкий набір команд (що мають назву командлети), призначених для управління функціонуванням інформаційної системи, виконання задач системного адміністрування (зокрема, управління процесами, службами, реєстром), виконання основних операцій з файловою системою тощо. Працюючи з Windows PowerShell, користувач також може використовувати команди операційної системи MS-DOS, засоби для роботи з об’єктами WMI та COM тощо.

Після ознайомлення з основними принципами роботи у середовищі Windows PowerShell (зокрема, призначенням вбудованих команд, засобів для роботи з процесами, службами, системними змінними та функціями, фільтрування виведення результатів виконання команд та ін.) студентів доцільно ознайомити з технологією розробки сценаріїв. Зокрема, розглянути такі питання:

- «політики» виконання сценаріїв (restricted, allsigned, remotesigned, unrestricted);
- типи даних (зокрема, int, float, long, double, decimal, string);
- надання значень змінним ($*назва змінної=значення змінної*);
- типи змінних (глобальні, локальні змінні, змінні сценарію, «приватні» змінні);
- оператори для роботи з числовими змінними (зокрема, $a+=$b, $a-=$b, $a*=$b, $a/=$b, $a%=$b, $a++, $a--, ++$a, --$a);
- оператори для роботи з символьними змінними (зокрема, ==, +, *, -Join, -Split);
- логічні оператори (-and, -or, -xor, -not, !);
- оператори порівняння (-eq, -ne, -lt, -le, -gt, -ge);
- оператори введення, виведення (read-host, write-host);
- оператори розгалуження (if, if…else, if…elseif … else);
- оператор вибору (swith);
- циклічні оператори (while, do…while, do … until, for, foreach);
- оголошення та виклик функцій користувача;
- виклик допоміжних сценаріїв;
- запуск сценаріїв з командного рядка Windows PowerShell тощо.

Доцільно звернути увагу студентів на те, що запуск сценаріїв з командного рядка Windows PowerShell здійснюється за допомогою команд *.\назва_сценарію* або *./назва_сценарію* (наприклад: .\script.ps1 або ./script.ps1) за умови, якщо каталог зі скриптами є поточною директорією та встановлена відповідна політика виконання сценаріїв (наприклад, remotesigned або unrestricted). У іншому випадку сценарії виконуватися не будуть. Для виконання сценаріїв з будь-якого робочого

каталогу можна включити директорію зі скриптами на час поточної сесії використання оболонки до переліку каталогів швидкого пошуку. Це робиться шляхом модифікації змінної оточення $env, наприклад: $env:path += “;C:\users\user1\scripts”.

Пояснення реалізації базових алгоритмічних структур засобами мови сценаріїв Windows PowerShell доцільно здійснювати на нескладних прикладах для розуміння призначення вказівок, аналізуючи результати їх покрокового виконання. У табл. 1 подано приклади використання операторів повторення Windows PowerShell.

Таблиця 1

Циклічні оператори	Приклад	Результат виконання

while (умова**) {**
оператори
}
	`$m=1`
	`while ($m -lt 10) {`
	`write-host $m`
	`$m=$m*2`
	`}`
		1
		2
		4
		8

do {
оператори
}
while (умова**)**
	`$a=5`
	`do {$a++`
	`write-host $a`
	`}`
	`while ($a -lt 9)`
		6
		7
		8
		9

do {
оператори
}
until (умова**)**
	`$b=5`
	`do {`
	`write-host $b`
	`$b--`
	`}`
	`until ($b -le 1)`
		5
		4
		3
		2

for (початкове значення**;** умова**;** крок**)**
{
оператори
}
	`for ($n=10; $n -gt 1;`
	`$n=$n-2)`
	`{`
	`write-host $n`
	`}`
		10
		8
		6
		4
		2

foreach (ім’я змінної **in** назва масиву**)**
{
оператори
}
	`$numb=5,-2,1.5,0,8,0,-29,13`
	`foreach ($i in $numb)`
	`{`
	`if ($i -gt 0) {`
	`write-host $i`
	`}`
	`}`
		5
		1.5
		8
		13

Для підвищення рівня розуміння студентами матеріалу доцільно провести порівняльні характеристики операторів, що розглядаються. Зокрема, можна обговорити зі студентам відмінності у використанні циклічних операторів за такими критеріями, як: кількість повторень; виконання операторів в залежності від логічного значення умови, що перевіряється; спосіб зміни значень лічильника циклу тощо. Наприклад, при використанні циклічного оператора while оператори тіла циклу можуть не виконатися жодного разу, якщо умова, що перевіряється, хибна, на відміну від циклів do…while, do…until, де істинність умови перевіряється після виконання операторів тіла циклу. Значення лічильника у циклі for змінюється автоматично, відповідно до встановленого кроку; у циклі foreach він набуває значень зі списку; для всіх інших типів циклів зміна значень лічильника відбувається у тілі циклу у результаті виконання відповідного оператора.

Достатньо ефективним з педагогічної точки зору у процесі оволодіння студентами особливостями реалізації базових структур алгоритмів під час створення сценаріїв засобами Windows PowerShell є використання методу відкритих програм. Студенти отримують код сценарію, призначення якого їм потрібно з’ясувати. Вони мають здійснити тестування сценарію для різних наборів вхідних даних та модифікувати його (наприклад, замінити цикл while циклом for або циклом until). Для перевірки засвоєння студентами синтаксису написання скриптів засобами Windows PowerShell у коді сценаріїв, що подаються, можуть міститися умисно допущені помилки. У цьому разі, студенти мають відлагодити код сценарію.

Після засвоєння студентами основних принципів написання сценаріїв доцільно запропонувати їм складніші завдання з використанням вивчених раніше команд Windows PowerShell, а також завдання професійного характеру.

Приклад 1. Створити сценарій, призначений для створення текстового файлу, ім’я якого вводить користувач у відповідь на запрошення. Якщо файл із заданим ім’ям уже існує, вивести відповідне повідомлення.

```
clear-host                                                  # очищення екрану
$f=read-host("Введіть назву файлу ")      # введення значення змінної $f
if ((Get-ChildItem|Where-object {$_.name -eq $f })) {`
                                  # перевірка  умови існування файлу з іменем $f
  write-host "Файл з таким ім'ям вже існує "} # виведення повідомлення
else {
  $v=read-host("Введіть вміст файлу");  # зчитування значення змінної $v
  ni -Name $f -ItemType file -Value $v }  # створення файлу із вмістом,
                                                  що зберігається у змінній $v
```

Приклад 2. Створити сценарій, що призначений для визначення кількості процесів у системі, на які відводиться більше 1MB пам'яті.

```
$i=0                                    # присвоєння значення 0 змінній $i
foreach ($s in get-process){`
                          # змінна $s набуває значення зі списку процесів
if ($s.WS -gt 1024) {`
                    # перевірка умови надання процесу пам'яті більше 1 MB
$i++}}                              # збільшення значення змінної $i на 1
write-host $i                                  # виведення значення змінної $i
```

Приклад 3. Описати функцію, призначену для виведення переліку імен об'єктів (файлів та підкаталогів) поточного каталогу, що були модифіковані після дати, що вказується користувачем як фактичні параметри. Результатом виконання функції має бути перелік назв об'єктів та відомостей про час їх останньої зміни.

```
function Get-NewFiles                   # заголовок функції
 { param ($d,$m,$y)                       # оголошення формальних параметрів
   $start = Get-Date  -Month $m  -Day $d  -Year $y
# надання змінній $start значення дати, що вводить користувач
   foreach ($f in Get-ChildItem|where {$_.LastWriteTime -gt $start}) {`
#  змінна $f набуває значення зі списку об'єктів поточного каталогу, що були
# змінені після введеної дати
   write-host ($f.name, $f.LastWriteTime)}
# виведення на екран відомостей про об'єкти та час їх останньої модифікації
 }
 Get-NewFiles -d 8 -m 12 -y 2014                 # виклик функції
```

У останньому прикладі продемонстровано створення та виклик функції користувача. Розуміння цього питання (зокрема, відмінностей між формальними та фактичними параметрами функції) має важливе значення для подальшого опанування студентами мовами програмування.

Для розробки сценаріїв досить зручно використовувати програму PowerShell Ise. Основними компонентами робочого вікна програми PowerShell Ise (Рис.1) є: головне меню, панель інструментів, поле введення команд та поле виведення результатів (аналогічно до командного рядка Windows PowerShell), поле розробки сценаріїв, вкладки для одночасної роботи з кількома задачами (сценаріями).

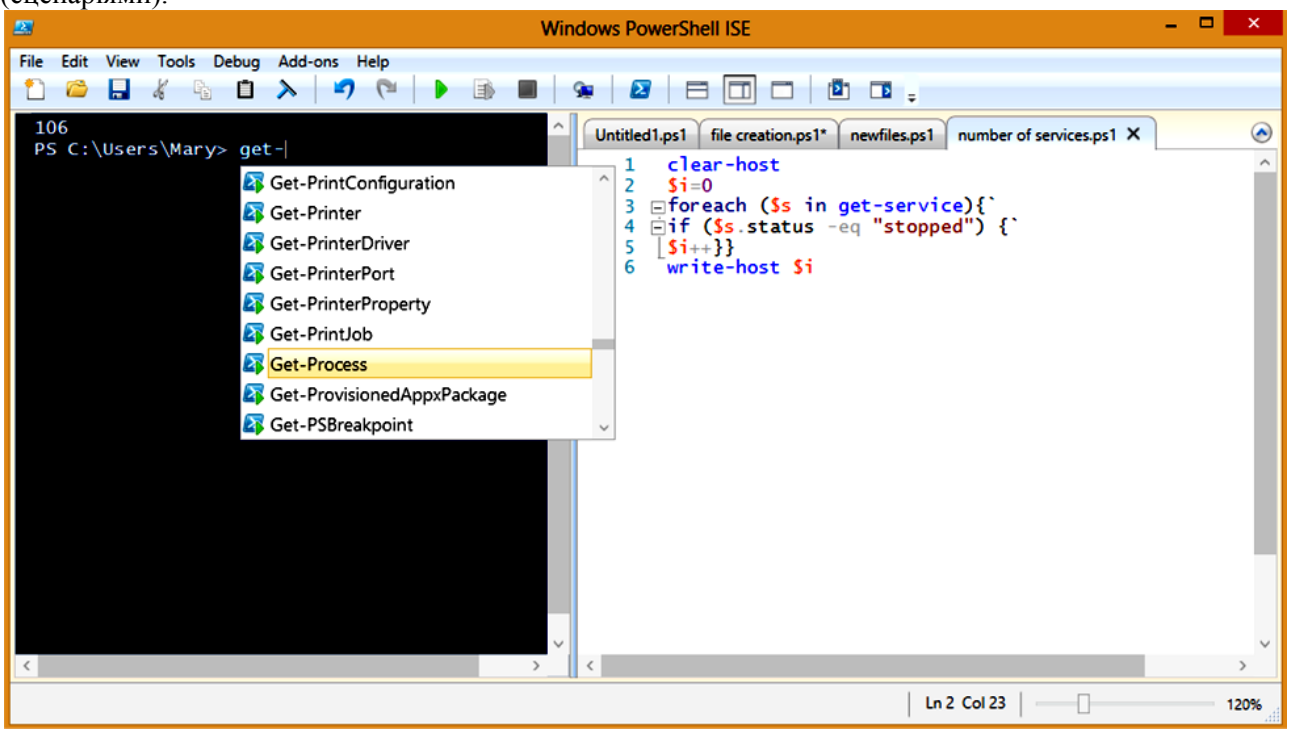

Рис. 1

Перевагами використання даного середовища для розробки сценаріїв, у порівнянні з використанням звичайних текстових редакторів, є наявність засобів для запуску, налагодження сценаріїв, кольорової розмітки синтаксису, виведення контекстної довідки при введенні операторів та команд тощо. Як показує практика, студенти швидко адаптуються до роботи у середовищі програми PowerShell Ise, зменшується кількість помилок при написанні сценаріїв, налагодження коду займає менше часу.

Слід зазначити, що в процесі створення сценаріїв засобами Windows PowerShell використовуються також принципи об'єктно-орієнтованого підходу у програмуванні. Мова сценаріїв Windows PowerShell з точки зору її синтаксичних особливостей та ключових слів, подібна до мови програмування C#, що є гарною пропедевтикою для її вивчення.

Вивчення командного інтерпретора bash операційної системи Linux [6] доцільно здійснювати за такими самими етапами, як це здійснювалось для Windows PowerShell:

1) розгляд вбудованих команд bash (зокрема, ls, cd, mkdir, touch, echo, cat, mv, cp, less, find та ін.);
2) написання сценаріїв (bash - скриптів):
   a) розгляд синтаксичних особливостей реалізації базових алгоритмічних структур в процесі створення сценаріїв;
   b) розв'язування задач, зокрема професійного характеру.

Оскільки принципи створення сценаріїв засобами Windows PowerShell та bash подібні, то можна вдатися до аналогії у процесі навчання деяких питань теми. Зокрема, як і у Windows PowerShell, запустити сценарій у bash можна за допомогою команди *./назва_сценарію* (наприклад *./script.sh*), якщо каталог зі скриптами є поточною директорією. При цьому доцільно звернути увагу студентів на те, що при роботі у середовищі bash користувач має заздалегідь передбачити можливість виконання файлу-сценарію (за допомогою команди: *chmod +x назва_файлу*), подібно до того, як встановлюється відповідна «політика» виконання сценаріїв у Windows PowerShell.

Слід наголосити також і на відмінностях у розробці сценаріїв засобами командних інтерпретаторів операційних систем Windows та Linux. При цьому, спочатку доцільно розглянути нескладні приклади застосування базових алгоритмічних структур для з'ясування відмінностей між використанням операторів одного типу. Наприклад, у табл.2 продемонстровано використання вказівок повторення для обчислення значення 5!. Легко бачити, що у bash використовується тільки 3 циклічні оператори: while, until та for.

Таблиця 2

Циклічні оператори	Приклад	Результат виконання

while [умова]
do
операторі
done
	`$a=1; $f=1`
	`while [ $a -le 5 ]`
	`do`
	`  $f=$f*a`
	`  $a=$a+1`
	`done`
		160

until [умова]
do
операторі
done
	`$a=1; $f=1`
	`until [ $a -ne 5 ]`
	`do`
	`  $f=$f*a`
	`  $a=$a+1`
	`done`
		160

for ім'я змінної **in** список
do
операторі
done
	`$f=1`
	`for $a in 1 2 3 4 5`
	`do`
	`  $f=$f*a`
	`  $a=$a+1`
	`done`
		160

З метою застосування набутих знань студентам доцільно запропонувати систему завдань для самостійного розв'язування. Окрему увагу слід приділити використанню операторів для роботи із рядками, зокрема:

- string1 = string2 – перевірка чи рядки string1 та string2 однакові;
- ${#string} – визначення довжини рядка string;
- ${string:0:3} – виокремлення трьох перших символів рядка string;
- ${string##*.} – вилучення найдовшого підрядка, що співпав із шаблоном «*.»;
- ${string#*.} - вилучення першого підрядка, що співпав із шаблоном «*.».

Приклад 4. Розробити сценарій, призначений для визначення типів файлів поточного каталогу за вказаним у їх назвах розширеннях.

```
#!/bin/bash            # початок сценарію
for filename in `ls`   # змінна filename набуває значень зі списку файлів
                            # поточного каталогу
do
   ext=${filename##*.}                 # виокремлення розширення файлу
   case "$ext" in                      # використання оператора вибору
         sh) echo "$filename - це сценарій bash";;
         txt) echo "$filename – текстовий файл";;
         c) echo "$filename – файл з с кодом ";;
         *) echo "невідомий тип файлу";;
   esac
 done
```

Важливо розглянути зі студентами питання розробки та використання сценаріїв для розв'язування завдань професійного характеру. Таким чином, розкривається практична значущість матеріалу, у студентів підвищується мотивація до навчання та набуття відповідних компетентностей. Педагогічно доцільно у цьому випадку застосувати метод демонстраційних прикладів. У процесі їх аналізу студенти засвоюють основні етапи розробки сценарію, використання базових структур алгоритмів, застосування методу послідовного уточнення алгоритму.

Приклад 5. Розробити сценарій для перевірки з'єднання машини користувача з іншими комп'ютерами локальної мережі.

```
#!/bin/bash
list=("192.168.1.10" "192.168.1.11" "192.168.1.12" "192.168.1.13")
for ip in $list do
      ping -c 1 $ip
      if [ $? -eq 0 ]; then
            echo "Комп'ютер з IP-адресою $ip під'єднано"
      else
            echo "Комп'ютер з IP-адресою $ip не відповідає"
      fi
done
```

Приклад 6. Розробити сценарій, призначений для створення резервної копії файлів, що були створені/модифіковані у певний день (дату вводить користувач) та надсилання їх на сервер у створений для користувача директорій.

```
echo "Enter the date (yyyy-mm-dd)"
read $d
listfile=`find ~ -type f -newermt $d`
mkdir ~/datedir
for i in $listfile; do
   cp $i ~/datedir/
done
tar czf datedir.tar.gz ~/datedir/
scp datedir.tar.gz student1@server_name:/remote/directory1
```

Таким чином, навчання студентів запропонованої технології розробки сценаріїв засобами Windows PowerShell та bash сприятиме усуненню прогалин у їх знаннях з основ алгоритмізації та програмування, формуванню у них відповідних компетентностей та готовності до вивчення мов програмування. У процесі вирішення професійних завдань студенти також набувають компетентностей щодо управління роботою інформаційної системи, що є компонентами системи їхніх інформатичних компетентностей.

Список використаних джерел

1. Жалдак М. І. Формування системи інформатичних компетентностей майбутніх учителів інформатики у процесі навчання в педагогічному університеті / Мирослав Жалдак, Юрій Рамський, Марина Рафальська // Вища школа. – 2009. – №10. – С. 44-52.
2. Рамський Ю. С. Зміни в професійній діяльності вчителя в епоху інформатизації освіти / Ю. С. Рамський // Науковий часопис НПУ імені М. П. Драгоманова. Серія №2. Комп'ютерно-орієнтовані системи навчання : зб. наук. праць / Редрада. – К. : НПУ імені М.П. Драгоманова, 2007. – №5(12). – С. 10-12.
3. Раков С. А. Сучасний учитель інформатики: кваліфікація та вимоги / С. А. Раков // Комп'ютер у школі та сім'ї. – 2005. – №5. – С. 35-38.

4. Зимняя И. А. Педагогическая психология : учебник для вузов / И. А Зимняя – М. : Логос, 1999. – 384 с.
5. Mastering-PowerShell. With: Dr. Tobias Weltner, PowerShell MVP. [Electronic resource]. – Mode of access : http://powershell.com/Mastering-PowerShell.pdf
6. Bash Guide for Beginners. [Electronic resource]. – Mode of access : http://www.tldp.org/LDP/Bash-Beginners-Guide/Bash-Beginners-Guide.pdf

Кобильник Т.П., Вдовичин Т.Я.
Дрогобицький державний педагогічний університет імені Івана Франка

Використання сервісу Wolfram|Alpha для розв'язування задач інтегрального числення функції однієї змінної

Зміст педагогічної освіти з напряму підготовки «Інформатика*» передбачає фундаментальну, психолого-педагогічну, методичну, інформаційно-технологічну, практичну і соціально-гуманітарну підготовку педагогічних і науково-педагогічних працівників.

Зміст фундаментальної підготовки передбачає вивчення теоретичних основ спеціальності згідно з вимогами до рівня теоретичної підготовки педагогічного працівника відповідного профілю і базується на новітніх досягненнях науки [3].

Тому випускники педагогічних університетів повинні володіти набором фундаментальних знань в галузі комп'ютерних наук, що дозволяло б швидко оволодівати сучасними комп'ютерними технологіями. Інформаційні технології та програмування базуються на класичних математичних дисциплінах, тому студентам напряму підготовки «Інформатика*» необхідні глибокі знання з математики.

Фундаментом математики є математичний аналіз. Основою математичного аналізу є диференціальне та інтегральне числення.

У навчанні студентів разом з іншими мережними технологіями особливу роль відіграють такі сервіси як web-СКМ. Це пояснюється такими причинами:

1) економічність: комерційні СКМ - дорогі програмні продукти, при цьому у педагогічному університеті використовуються, як правило, для навчання тільки незначної кількості дисциплін;
2) кросплатформенність: перехід до web-інтерфейсу – це найпростіший шлях перенесення можливостей використання СКМ на всі існуючі платформи, в тому числі і мобільні.

Враховуючи наведені причини, основні виробники популярних комерційних СКМ, зокрема MapleSoft (розробник Maple) та Wolfram Research (Mathematica) створили та підтримують online-сервіси MapleNet та webMathematica і Wolfram|Alpha відповідно.

Стаття присвячена аналізу можливостей використання web-сервісу Wolfram|Alpha для розв'язування задач інтегрального числення функцій однієї змінної.

Подібним проблемам присвячено ряд досліджень різних авторів. Зокрема в роботах Ю.В. Триуса розглядаються проблеми впровадження в навчальний процес ВНЗ web-СКМ SAGE та Wolfram|Alpha [7-8]. У статті Ю.В.Горошка та Д.А. Покришеня [1] наведено загальну характеристику бази знань Wolfram|Alpha та описано можливості використання до розв'язування окремих математичних задач. У роботі Д.А.Покришеня та Є.Ю. Носенко [4] розглянуто з коротким описом кілька програмних продуктів різних виробників з різною методичною направленістю та наведено приклади використання математичних задач у вивченні інформатики за допомогою програмних продуктів GRAN1, Wolfram|Alpha, Microsoft Mathematics 4.0. Поряд з тим існує і web-СКМ Sage, у якій реалізовано підтримку інтерфейсів, в тому числі і до комерційних СКМ. Про систему Sage є достатньо багато публікацій, зокрема цю проблематику досліджували С.О. Семеріков та його учні [5-6].

На особливу увагу заслуговує навчальний посібник [2], у якому поряд з теоретичними відомостями наведено приклади застосування комп'ютерних засобів математики, зокрема Gran1, Maxima, MathCAD, до розв'язування задач інтегрального числення функцій однієї змінної.

У педагогічному університеті інтегральне числення функцій однієї змінної студентами напряму підготовки «Інформатика*» вивчається на першому курсі. Студенти, як правило, ще не вміють працювати з математичними пакетами для символьних перетворень, зокрема СКМ Mathematica, Maple та іншими. Як правило, такі СКМ є «вибагливими» до синтаксису, на відміну від сервісу Wolfarm|Alpha. Тому якщо є доступ до мережі Internet, для супроводу навчання бакалаврів інформатики зручно використовувати web-сервіс Wolfram|Alpha. Цей сервіс заснований на опрацюванні природної мови (поки тільки англійської), великій базі знань та алгоритмів. Наприклад, система однаково «сприйме» запит «integrate 1/x» та «int 1/x». Відповіддю на такий запит буде невизначений інтеграл для функції $\frac{1}{x}$ (рис. 1, а та рис. 1, б)