МІНІСТЕРСТВО ОСВІТИ І НАУКИ, МОЛОДІ ТА СПОРТУ УКРАЇНИ
КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БУДІВНИЦТВА І АРХІТЕКТУРИ

Об’єктно-орієнтоване програмування

Методичні вказівки
до виконання індивідуального завдання
для студентів, які навчаються за напрямом підготовки
6.050101 «Комп’ютерні науки»
спеціалізації «Інформаційні управляючі системи і технології»

Київ 2011

ББК 32.937.26-01
О-13

Укладач Г.В. Красовська, канд. техн. наук, доцент

Рецензент О.В. Федусенко, канд. техн. наук, доцент

Відповідальний за випуск В.Б. Задоров, канд. техн. наук, професор

Затверджено на засіданні кафедри інформаційних технологій, протокол №11 від 17 січня 2011 року.

Видається в авторській редакції.

Об'єктно-орієнтоване програмування: методичні вказівки до
О-13 виконання індивідуального завдання і курсової роботи / Уклад. Г.В. Красовська. – К.: КНУБА, 2011. – 24 с.

Призначено для студентів, які навчаються за напрямом підготовки 6.050101 «Комп'ютерні науки» спеціалізації «Інформаційні управляючі системи і технології» для використання в процесі виконання індивідуального завдання.

Містять загальні положення, порядок виконання роботи, варіанти індивідуального завдання, методичні вказівки до виконання роботи, список літератури, додатки.

ЗМІСТ

Загальні положення

Виконання індивідуального завдання з дисципліни «Об'єктно-орієнтоване програмування» дозволяє студентам набути практичних навичок аналізу та формалізації задачі зі створення класів програмних об'єктів, розробки, тестування та налагодження програм в середовищі візуального програмування Delphi та C++, сприяє закріпленню та поглибленню студентами теоретичних знань з основних розділів дисципліни «Об'єктно-орієнтоване програмування», що дозволить студентам підготуватись до успішного виконання курсової роботи в четвертому семестрі.

Протягом третього навчального семестру студентом виконується 1 індивідуальне завдання, на виконання якого учбовим планом відведено 8 тижнів.

Теми індивідуальних завдань охоплюють чотири розділи дисципліни «Об'єктно-орієнтоване програмування»: «Базові поняття об'єктно-орієнтованого програмування. Об'єктна модель мови Delphi Pascal та C++», «Інкапсуляція», «Спадковість», «Поліморфізм».

Порядок виконання індивідуального завдання

Варіант індивідуального завдання обирається студентом зі списку варіантів, що наведений нижче у методичних вказівках. Номер варіанта індивідуального завдання відповідає номеру студента у списку академічної групи.

Під час виконання індивідуального завдання:

1. розробка ієрархії класів програмних об'єктів:
   - 1.1. розробка батьківського класу:
     - 1.1.1. визначення полів батьківського класу, опис призначення полів;
     - 1.1.2. визначення методів та постановка задач методів батьківського класу;
   - 1.2. розробка дочірнього класу:
     - 1.2.1. визначення, чи необхідно додавати в дочірньому класі до успадкованих полів від батьківського класу нові поля. Якщо так – описати їх призначення;
     - 1.2.2. визначення, чи необхідно додавати в дочірньому класі до методів батьківського класу нові, чи можна просто змінити

реалізацію батьківського метода. Для методів дочірнього класу розробити постановку задач. Якщо в дочірньому класі перевизначаються методи батьківського класу, необхідно:

- в батьківському класі описати відповідний метод як віртуальний;
- в дочірньому класі перевизначити цей метод директивою override.

2. розробка програмного застосування згідно до завдання на Delphi та C++;
   2.1. опис визначених класів на Delphi та C++;
   2.2. розробка програмного інтерфейсу користувача, опис сценарію прецеденту;
   2.3. створення екземплярів класів та демонстрація їх роботи;
3. тестування роботи програми;
4. оформлення звіту з індивідуального завдання та його захист.

Індивідуальне завдання вважається закінченим, якщо отриманий результат по всіх тестових прикладах роботи програми, оформлений звіт згідно з вимогами, що наведені в методичних вказівках. Під час здачі індивідуального завдання студент вільно орієнтується в теоретичному матеріалі, правильно відповідає на запропоновані запитання.

Кожного тижня студент доповідає викладачу про хід виконання роботи.

Методичні рекомендації до виконання роботи

Розглянемо виконання індивідуального завдання на конкретному прикладі: визначити клас для дійсних чисел. Поля класу - ціла та дробова частини числа. Методи класу - виведення значення числа на екран.

Розробка ієрархії класів програмних об'єктів. Розробка батьківського класу TVariable

В якості батьківського розглянемо клас для цілочисельної змінної, що детально описаний на сторінці 13 в темі «Інкапсуляція» електронного конспекту лекцій з дисципліни «Об'єктно-орієнтоване програмування» http://org.knuba.edu.ua/mod/resource/view.php?id=5603.

Визначення полів батьківського класу TVariable, опис призначення полів

Поля класу TVariable наведені в табл. 1.

Таблиця 1

Поля батьківського класу TVariable

Delphi	C++	Опис призначення
FValue: integer;	int fvalue;	директива видимості: приватне; містить значення цілочисельної змінної;
FName:string [63];	char* fname;	директива видимості: приватне; містить ім'я цілочисельної змінної. Довжина імені не більша за 63 символи, відповідає правилам визначення ідентифікаторів мов програмування.

Визначення методів та постановка задач методів батьківського класу TVariable

Методи класу TVariable наведені в табл. 2.

Таблиця 2

Методи батьківського класу TVariable

Delphi	C++	Опис
function GetName :string;	char* getname();	*Директива видимості*: захищене; *Вхідні дані*: не має *Вихідні дані*: ім'я змінної *Призначення*: метод для зчитування значення поля FName; в Delphi використовується властивістю Name.
function GetValue :integer;	int getvalue();	*Директива видимості*: захищене; *Вхідні дані*: не має *Вихідні дані*: значення цілочисельної змінної *Призначення*: метод для зчитування значення поля FValue; в Delphi використовується властивістю Value.

Продовження табл. 2

Delphi	C++	Опис
procedure SetName (N:string);	void setname (char* n);	*Директива видимості*: захищене; *Вхідні дані*: нове значення цілочисельної змінної *Вихідні дані*: не має *Призначення*: метод для запису нового значення в поле FName; в Delphi використовується властивістю Name.
procedure SetValue (V:integer);	void setvalue (int v);	*Директива видимості*: захищене; *Вхідні дані*: нове значення цілочисельної змінної *Вихідні дані*: не має *Призначення*: метод для запису нового значення в поле FValue; в Delphi використовується властивістю Value.
constructor Create (V:integer; N:string);	tvariable (int v, char* n);	*Директива видимості*: загальнодоступне; *Вхідні дані*: нове значення цілочисельної змінної та її ім'я *Вихідні дані*: не має *Призначення*: конструктор класу також проводиться ініціалізація полів значеннями, що передані як параметри конструктора
function ShowFields :string;	char* showfields();	*Директива видимості*: загальнодоступне; *Вхідні дані*: не має *Вихідні дані*: рядок з поточними значеннями полів *Призначення*: формує з полів класу FName та FValue і повертає як результат рядок такого виду: *ім'я змінної **:** значення*

Закінчення табл. 2

Delphi	C++	Опис
property Value:integer read GetValue write SetValue; property Name:string read GetName write SetName;	немає	*Директива видимості*: загальнодоступне; *Призначення*: властивості для доступу до полів FName та FValue

Розробка ієрархії класів програмних об'єктів. Розробка дочірнього класу

Дочірній клас TFloatVariable для змінних дійсного типу буде відрізнятися від батьківського наявністю дробової частини числа. Отже, необхідно до полів батьківського класу додати поле для збереження дробової частини.

Важливою рисою для дійсних значень є порядок числа (кількість знаків після точки, що достовірно зберігаються). Доцільно ввести також поле, що буде зберігати значення порядку для заданого дійсного. Це поле буде використовуватися при формуванні дробової частини числа.

Визначення полів дочірнього класу TFloatVariable, опис призначення полів

Поля класу TFloatVariable наведені в табл. 3.

Таблиця 3

Поля батьківського класу TVariable

Delphi	C++	Опис призначення
FDecimal : real;	float fdecimal;	директива видимості: приватне; містить значення цілочисельної змінної;
Poriadok:integer;	int poriadok;	директива видимості: приватне; містить ім'я цілочисельної змінної. Довжина імені не більша за 63 символи, відповідає правилам визначення ідентифікаторів мов програмування.

Визначення методів та постановка задач методів дочірнього класу TFloatVariable

В зв'язку з тим, що в дочірньому класі визначене поле з дробовою частиною, необхідно переробити реалізацію метода ShowFields, тому що необхідно виводити як цілу так і дробову частину числа. Щоби надати змогу дочірньому класу перевизначити реалізацію метода батьківського класу необхідно:

- в батьківському класі описати метод ShowFields як віртуальний;
- в дочірньому класі описати цей метод директивою override.

Методи класу TVariable наведені в табл. 4. Остаточний вигляд класів можна продивитися в додатках 1-2.

Таблиця 4

Методи батьківського класу TVariable

<table>
<tr><th>Delphi</th><th>C++</th><th>Опис призначення</th></tr>
<tr><td>function GetDec :integer;</td><td>int getdec();</td><td>Директива видимості: захищене;
метод для зчитування значення поля FDecimal;
в Delphi використовується властивістю Decimal.</td></tr>
<tr><td colspan="3">Математична постановка задачі:
Вхідні дані: не має
Вихідні дані: послідовність цифр, що становить дробову частину числа
Опис призначення: формує як ціле послідовність цифр, що становить дробову частину числа (з поля класу FDecimal).
Алгоритм роботи:
(Схема алгоритму обов'язково наводиться в звіті на окремому рисунку)
1. D:=FDecimal.
2. Для i:=1, poriadok виконувати D:=D*10.
3. Як результат повернути D.</td></tr>
<tr><td>procedure SetDec (D:integer);</td><td>void setdec (int d);</td><td>Директива видимості: захищене;
метод для запису значення поля FDecimal;
в Delphi використовується властивістю Decimal.</td></tr>
</table>

Закінчення табл.4

Delphi	C++	Опис призначення
Математична постановка задачі: *Вхідні дані:* послідовність цифр, що становить дробову частину числа *Вихідні дані:* не має *Опис призначення:* задану як ціле послідовність цифр переводить в дійсне значення, що становить дробову частину числа (поле класу FDecimal). Під час переведення визначається також порядок числа, у даному випадку – кількість цифр в дробовій частині числа. *Алгоритм роботи:* ***(Схема алгоритму обов'язково наводиться в звіті на окремому рисунку)*** 4. Якщо значення D>0, встановити poriadok:=0, FDecimal:=d, інакше FDecimal:=0. Вихід. 5. Доки FDecimal >= 1 виконувати: FDecimal:= FDecimal /10; poriadok:=poriadok+1;		
constructor Create (V:integer; D:integer; N:string);	tfloatvariable (int v, int d, char* n);	*Директива видимості*: загальнодоступне; *Вхідні дані*: нове значення цілочисельної змінної та її ім'я *Вихідні дані*: не має *Призначення*: конструктор класу також проводиться ініціалізація полів значеннями, що передані як параметри конструктора
function ShowFields:string; override;	char* showfields();	*Директива видимості*: загальнодоступне; *Вхідні дані*: не має *Вихідні дані*: рядок з поточними значеннями полів *Призначення*: перевизначає метод ShowFields, що успадкований від батьківського класу TVariable; формує з полів класу FName, FValue, FDecimal і повертає як результат рядок такого виду: *ім'я змінної : значення . дробова частина*
property Decimal:integer	немає	*Директива видимості*: загальнодоступне;

read GetDec write SetDec;		*Призначення*: властивості для доступу до поля FDecimal.

Після цього можна:

- побудувати UML – діаграму класів;
- в середовищі Delphi та C++ створити нові проекти, додати до них нові модулі і описати в них ієрархію класів, що була утворена.

Розробка програмного інтерфейсу користувача, опис сценарію прецеденту

Після того як буде завершений опис класів і виправлені всі помилки необхідно продумати, які елементи інтерфейсу з користувачем будуть використанні на формі для:

- створення екземплярів батьківського і дочірнього класів;
- введення (редагування) значення полів;
- відображення поточного значення полів класу;
- виклику методів класу для демонстрації їх роботи.

Конструктори класів в Delphi можна викликати при створенні форми з обробника OnCreate, тому спеціальні елементи управління для його виклику не потрібні.

В C++ екземпляри класів доцільно створювати в файлі форми на зовнішньому рівні.

Для введення значень полів батьківського класу знадобляться два компонента Edit, а для дочірнього - три.

Для виведення рядків з назвою та значенням змінних (метод ShowFields) можна використати мітки (Label).

Для виклику метода ShowFields можна використати просту кнопку Button, задавши її обробник OnClick.

Вибір того чи іншого з інтерфейсних елементів для відображення “начинки” класу не обмежується, треба тільки не порушувати умов функціонального призначення компонентів.

В результаті розробки програмного інтерфейсу головне вікно набуває вигляд, що поданий на рисунку 1.

Після цього можна:

- описати сценарій прецеденту;

Створення екземплярів класів та демонстрація їх роботи

Для того щоб створити екземпляри батьківського і дочірнього класів необхідно:

1. під’єднати модулі, де описані ці класи до модуля головної форми;
2. описати екземпляри класів;
3. для створення екземплярів класів Delphi в обробнику OnCreate форми викликати конструктори класів, задавши якісь початкові значення полів:

   X:=TVariable.Create(0,'');

   Y:=TFloatVariable.Create(0, 0 , '');
4. задати для елементів управління на формі всі необхідні обробники подій згідно їх призначення.

Остаточний варіант тексту програми на Delphi можна переглянути у додатках 1-3.

Після цього можна:

- побудувати UML – діаграму компонентів;
- описати призначення файлів, що входять до складу проектів (програмного застосування).

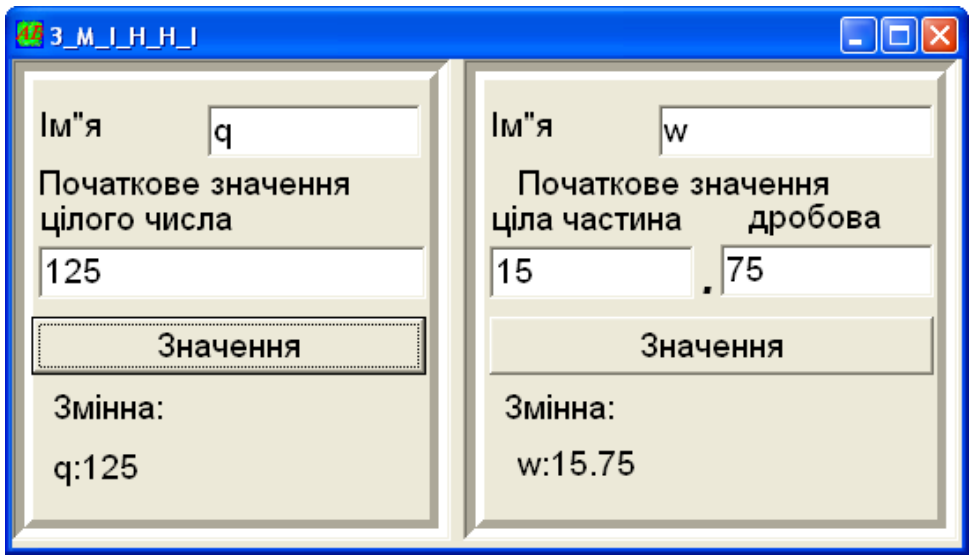

Рис. 1. Головне вікно програми

Тестування роботи програми

Під час тестування програми необхідно довести працездатність **всіх методів** батьківського і дочірнього класів. При цьому необхідно перевірити всі гілки розгалуження, якщо такі є в методі. І навести *розрахунки вручну*, якщо вони для метода існує *математична постановка задачі*.

Наприклад, під час перевірки правильності завдання ідентифікатору змінної в методі SetName необхідно розглянути такі випадки:

- довжина імені Length(N) = 0 – ім’я не задане;
- довжина імені Length(N) ≠ 0 – ім’я задане;
    - o довжина імені більша за 63 символи;
    - o довжина імені не більша за 63 символи і:
        - починається з цифри;
        - включає цифри в кінці або в середині;
        - містить пробіли;
        - містить знаки операцій;
        - містить знак підкреслення.

Оформлення звіту з індивідуального завдання та його захист

Звіт з індивідуального завдання включає:

1. титульний лист;
2. зміст;
3. завдання до роботи;
4. опис класів (UML – діаграма класів, опис полів та методів кожного класу);
5. опис програмного інтерфейсу користувача подається у вигляді опису прецеденту.
6. опис програмного застосування – UML-діаграма компонентів (окремо будується діаграма для Delphi та C++ застосувань), опис файлів, що входять до складу;
7. тестові розрахунки. Для перевірки правильності роботи програми для кожного розрахункового метода класу наводяться приклади відповідних розрахунків вручну та отримані результати роботи програми;
8. список використаної літератури;
9. додатки з роздруком текстів програмних модулів.

При оформленні звіту необхідно дотримуватися державних стандартів, що висуваються до оформлення технічної документації:

- звіт оформлюється на листах білого паперу формату А4;
- текст розташовується на листі, додержуючись таких розмірів полів: ліве – 2.5 см, праве – 2 см, верхнє та нижнє – 2.5 см;
- міжрядковий інтервал – 1.5;
- розмір шрифту (за умови машинного набору) – TimesNewRoman 14 пт або Arial 12 пт;
- форматування абзаців – по ширині з відступом 1,27 пт;

- розділи звіту (пункти та підпункти) повинні мати заголовки. Після заголовку крапка не ставиться, заголовки не підкреслюються. Після заголовку перед текстом пропускається один рядок.

- ілюстрації (рисунки, діаграми, схеми алгоритмів, роздрук екранних форм програмного застосування) розміщуються безпосередньо після тексту, де він вперше згадується або за браком місця на наступній сторінці. На всі ілюстрації мають бути посилання у тексті.

Ілюстрація позначається словом “Рисунок <номер> ” або “Рис. <номер> ”, яке разом з назвою ілюстрації розміщують безпосередньо **після** неї. Наприклад, «Рисунок 2. Структурна схема програмних модулів» ;

- таблицю розташовують безпосередньо після тексту, де є посилання на неї, або на наступній сторінці. Над таблицею по центру розташовується заголовок всієї таблиці у формі:

Таблиця < номер > – < назва таблиці >

Якщо таблиця розривається, то над частинами таблиці пишуть – “Продовження таблиці – < номер > ” .

- Формули та рівняння розташовуються безпосередньо після тексту, де є посилання на них. Вище та нижче кожного рівняння або формули залишається по одному вільному рядку. Формули і рівняння розташовуються по центру рядка та нумеруються. Номер ставиться з правого боку сторінки у дужках ().

Додаткам передує лист, на якому по центру розміщується заголовок “ДОДАТКИ”. Всі додатки нумеруються та супроводжуються заголовками виду: “Додаток 1. Роздрук тексту програмного модуля MyClass.pas”.

Для детального ознайомлення з вимогами до оформлення звітів дивіться стандарти «Єдиної системи конструкторської документації» та «Єдиної системи програмної документації».

Варіанти індивідуального завдання

Варіанти 1-7

Для варіантів 1-7 як батьківський використати клас TVariable, що описаний вище.

1. Визначити клас для комплексних чисел. Поля класу - дійсна

та комплексна частини числа. Методи класу - виведення значення числа на екран, додавання, віднімання, добуток комплексних чисел. Під час реалізації класів в С++ перевантажити відповіді оператори для цього класу.

2. Визначити клас для десяткових дробів. Поля класу - чисельник, знаменник та ціла частина дробі. Методи класу - додавання, віднімання, добуток, ділення, нормалізація дробі. Під час реалізації класів в С++ перевантажити відповіді оператори для цього класу.

3. Визначити клас для числа в заданому степені. Поля класу – значення числа, значення показника степені. Методи класу – зведення до степені, добуток та ділення степенів. Під час реалізації класів в С++ перевантажити відповіді оператори для цього класу.

4. Визначити клас для арифметичної прогресії. Визначити методи для обчислення і-го члена, виведення прогресії на екран, визначення суми заданої кількості членів прогресії. Під час реалізації класів в С++ перевантажити оператори присвоєння та додавання для цього класу.

5. Визначити клас для геометричної прогресії. Визначити методи для обчислення і-го члена, виведення прогресії на екран, визначення суми заданої кількості членів прогресії. Під час реалізації класів в С++ перевантажити оператори присвоєння та додавання для цього класу.

6. Визначити клас для ряду Фібоначчі прогресії. Визначити методи для обчислення і-го члена ряду, виведення ряду на екран, визначення суми заданої кількості членів ряду. Під час реалізації класів в С++ перевантажити оператори присвоєння та додавання для цього класу.

7. Визначити клас для змінної-рядка символів, що “знає” свою довжину та кількість слів, перевіряє, чи є він анаграмою. Поля класу: ім’я рядка, довжина рядка, вміст рядка. Під час реалізації класів в С++ перевантажити оператори присвоєння та додавання для цього класу.

Варіанти 8-16

Для класів плоских геометричних фігур за базовий батьківський клас взяти клас точки. Використати поля цього

класу як точку прив'язки під час зображення плоскої геометричної фігури на екрані.

8. Визначити клас для прямокутника. Поля класу – координати лівого верхнього та правого нижнього кутів. Методи – виведення зображення на екран, визначення периметру та площі.

9. Визначити клас для трикутника. Поля класу – координати вершин. Методи – виведення зображення на екран, визначення довжини сторін, периметру та площі.

10. Визначити клас для кола. Поля класу – координати центра та радіус. Методи – виведення зображення на екран, визначення довжини та площі.

11. Визначити клас для прямокутника. Поля класу – довжини сторін. Методи – виведення зображення на екран, визначення периметру та площі.

12. Визначити клас для ромба. Поля класу – довжини діагоналей (точка прив'язки на їх перетині). Методи – виведення зображення на екран, визначення довжини сторони, периметру та площі.

13. Визначити клас для квадрату. Поля класу – довжина сторони. Методи – виведення зображення на екран, визначення периметру та площі.

14. Визначити клас для квадрату. Поля класу – довжина діагоналі. Методи – виведення зображення на екран, визначення сторони, периметру та площі.

15. Визначити клас для прямокутного трикутника. Поля класу – довжини катетів (точка прив'язки – прямий кут). Методи – виведення зображення на екран, визначення гіпотенузи, периметру та площі.

16. Визначити клас для правильного багатокутника. Поля класу – кількість сторін (точка прив'язки – центр описаного кола). Методи – виведення зображення на екран (описане коло та одна сторона з зазначенням центрального кута), визначення центрального кута та площі.

Варіанти 17-21

За базовий батьківський клас взяти клас точки. Від нього утворити клас кола. Поля – координати центра, радіус. Методи –

визначення довжини та площі.

17. Визначити клас для кільця. Поля – два радіуси. Методи – визначення площі та довжин зовнішнього та внутрішнього кіл.

18. Визначити клас для еліпсу. Поля класу – Rx та Ry. Методи – визначення площі та довжини.

19. Визначити клас для конуса. Поля класу – радіус основи та висота. Методи – визначення площі поверхні та об'єму.

20. Визначити клас для циліндра. Поля класу – радіус основи та висота. Методи – визначення площі поверхні та об'єму.

21. Визначити клас для сектору. Поля класу – кут та радіус. Методи – визначення площі та довжини дуги.

Варіанти 22-27

Для класів шахових фігур визначити один загальний клас для будь-якої шахової фігури, в якому описана точка прив'язки, колір, ім'я файлу з графічним зображенням, метод для виведення зображення, метод для перевірки можливості переміщення (віртуальний абстрактний, перевизначається в нащадках). Для відображення шахової фігури на формі можна створити простеньке графічне зображення (наприклад, в ImageEditor або Paint), а потім вивести його на форму в компоненті Image (палітри Additional).

22. Визначити клас для шахової фігури - ладдя.

23. Визначити клас для шахової фігури - слон.

24. Визначити клас для шахової фігури - ферзь.

25. Визначити клас для шахової фігури - кінь.

26. Визначити клас для шахової фігури - король. Метод для рокіровки короля.

27. Визначити клас для шахової фігури - пішка.

Варіант 28

28. Визначити клас для квадратного рівняння. Поля класу – коефіцієнти рівняння. Методи – виведення рівняння на екран, визначення коренів. Описати дочірній клас – біквадратне рівняння.

СПИСОК ЛІТЕРАТУРИ

1. *Дарахвелидае П. Г., Марков Е. П.* Программирование в Delphi 7. – СПб.: БХВ-Петербург, 2003. – 784 с : ил.

2. *Дарахвелидзе П.Г., Марков Е.П.* Delphi – среда визуального программирования: СПб.: BHV-Санкт Петербург, 1996. – 352 с.

3. *Ларман К.* Применение UML и шаблоны проектирования: Уч. пос. – М.: Вильямс, 2001. – 496 с.

4. *Сван Том*, Основы программирования Delphi для WINDOWS 95. Пер. с англ. – К.: Диалектика, 1996. – 480 с., ил.

5. *Шилдт Г*. Самоучитель C++: Пер. с англ. – 3-е изд. – СПб.: БХВ-Петербург, 2003. – 688 с.

Роздрук тексту програмного модуля опису класу Variable_cl.pas

```
unit Variable_cl;

interface
Type
TVariable = class
private
        FValue: integer;
        FName:string [63];
protected
//методи для зчитування полів FName та FValue
function GetName :string;
function GetValue :integer;
//методи для запису полів FName та FValue
procedure SetValue (V:integer);
procedure SetName (N:string);
public
//властивості для доступу до  полів FName та FValue
property Value:integer read GetValue write SetValue;
property Name:string read GetName write SetName;

constructor Create (V:integer; N:string);
function ShowFields:string; virtual;
end;

Var
 X:TVariable;

implementation
uses Dialogs, SysUtils;

function TVariable.GetName: string;
begin
  Result:= FName;
end;

function TVariable.GetValue: integer;
begin
  Result := FValue;
end;

procedure TVariable.SetName(N: string);
 var
  i:byte;
  Digit, S_Alpha, B_Alpha:set of char;
  Good:boolean;
begin
```

```
Digit:=['0'..'9']; S_Alpha:=['a'..'z']; B_Alpha:=['A'..'Z'];
Good:=true; FName:='';
if Length(N)<>0 Then
begin
if Length (N)>=63 then
  begin
    ShowMessage ('Дуже довгий ідентифікатор!!!' );
    exit;
  end;
if (N[1] in Digit) then Good:=false
else
 for i:=1 to length (N) do
   if not                                  // якщо не
        ( (N[i] in S_Alpha) or (N[i] in B_Alpha) //  буква,
            or(N[i] in Digit) or (N[i]='_') ) then //цифра або знак підкреслення
   begin
    Good:=false;
    break;
   end;

  if Good then   FName:=N else ShowMessage ('Невірний ідентифікатор!!!');
end;
end;

procedure TVariable.SetValue(V: integer);
begin
  if FValue <> V then   FValue:=V;
end;

constructor TVariable.Create(V: integer; N: string);
begin
  Value:=V;
  Name:=N;
end;

function TVariable.ShowFields: string;
begin
 Result:=GetName + ':' + IntToStr(GetValue);
end;

end.
```

Роздрук тексту програмного модуля опису класу FloVar_cl.pas

```
unit FloVar_cl;

interface
uses variable_cl;
type
 TFloatVariable = class (TVariable)
   private
    FDecimal : real;
    Poriadok:integer;
   protected
    function GetDec :integer;
    procedure SetDec (D:integer);
   public
    constructor Create (V:integer; D:integer; N:string );
    function ShowFields:string;   override;
    property Decimal:integer read GetDec write SetDec;
 end;

var Y: TFloatVariable;

implementation
uses SysUtils;
{ TFloatVariable }

constructor TFloatVariable.Create(V: integer; D: integer; N: string);
begin
 inherited create(V,N);
 Decimal:=D;
end;

function TFloatVariable.GetDec: integer;
var i: byte;
      D:real;
begin
    d:=FDecimal;
    for i:=1 to poriadok do  d:=D*10;
    Result:= trunc(D);
end;

procedure TFloatVariable.SetDec(D: integer);
begin
if D>0 then
 begin
   poriadok:=0;     FDecimal:=d;
   while FDecimal >= 1 do
    begin
```

```
      FDecimal:= FDecimal /10;
      poriadok:=poriadok+1;
    end;
  end
else FDecimal:=0;
end;

function TFloatVariable.ShowFields: string;
begin
  Result:= inherited ShowFields +'.'+ IntToStr (Decimal);
end;


end.
```

Роздрук тексту програмного модуля Variable_fm.pas

```
unit Variable_fm;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, Variable_cl, ExtCtrls,  FloVar_cl;
type
  TForm1 = class(TForm)
      … … …
      //опис всіх елементів управління , що розташовані на формі
  end;
var
  Form1: TForm1;

implementation
{$R *.dfm}

procedure TForm1.FormCreate(Sender: TObject);
begin
 X:=TVariable.Create(0,'');
 Edit1.Text:=X.Name;
 Edit2.Text:=IntToStr(X.Value);

 Y:=TFloatVariable.Create(0, 0 , '');
 Edit3.Text:=Y.Name;
 Edit4.Text:=IntToStr(Y.Value);
 Edit5.Text:=FloatToStr(Y.Decimal);
end;

procedure TForm1.Button2Click(Sender: TObject);
begin
 Y.Name:=Edit3.Text;
 Y.Value:=StrToInt(Edit4.text);
 Y.Decimal :=STRToint(Edit5.text);

 Label10.Caption:=Y.ShowFields;
end;

procedure TForm1.Button3Click(Sender: TObject);
begin
 X.Name:=Edit1.Text;
 X.Value:=StrToInt(Edit2.text);

 Label4.Caption:=X.ShowFields;
end;

end.
```

Навчально-методичне видання

Об’єктно-орієнтоване програмування

Методичні вказівки
до виконання індивідуального завдання
для студентів, які навчаються за напрямом підготовки
6.050101 «Комп’ютерні науки»
спеціалізації «Інформаційні управляючі системи і технології»

Укладач КРАСОВСЬКА Ганна Валеріївна

Комп’ютерне верстання *О.В. Кириченка*

Підписано до друку Формат 60x84 1/16.
Ум. друк. арк. 1,39. Обл.-вид. арк. 1,5.
Тираж 40 прим. Вид. № 24/III-11. Зам. №
КНУБА, Повітрофлотський проспект, 31, Київ, Україна, 03680

E-mail: red-isdat@knuba.edu.ua

Видруковано в редакційно-видавничому відділі
Київського національного університету будівництва і архітектури

Свідоцтво про внесення до Державного реєстру суб’єктів видавничої справи
ДК № 808 від 13.02.2002 р.