МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ХЕРСОНСЬКИЙ ДЕРЖАВНИЙ ПЕДАГОГІЧНИЙ УНІВЕРСИТЕТ

КАФЕДРА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Методичні рекомендації до лабораторних робіт

з курсу *“Об’єктно-орієнтоване програмування”*

для спеціальностей:
7.010103 "ПМСО. Математика. Інформатика";
7.010103 "ПМСО. Фізика. Інформатика";
7.010103 "ПМСО. Математика та основи інформатики";
7.010103 "ПМСО. Фізика та основи інформатики";
7.010103 "ПМСО. Англійська мова та література. Спеціалізація: основи інформатики".
7.010103 "ПМСО. Хімія та основи інформатики";

Херсон - 2000

Методичні рекомендації до лабораторних робіт з курсу
“Об’єктно-орієнтоване програмування”

для спеціальностей:
7.010103 "ПМСО. Математика. Інформатика";
7.010103 "ПМСО. Фізика. Інформатика";
7.010103 "ПМСО. Математика та основи інформатики";
7.010103 "ПМСО. Фізика та основи інформатики";
7.010103 "ПМСО. Англійська мова та література. Спеціалізація: основи інформатики".
7.010103 "ПМСО. Хімія та основи інформатики";

Укладачі: Співаковський О.В. – канд. фіз.-мат. наук, доцент,
Зайцева Т.В. – старший викладач,
Сінько Ю.І. – старший викладач.
За ред. О.В. Співаковського.

Рецензенти: начальник відділу науково-дослідної та редакційної роботи Херсонського факультету Запорізького юридичного інституту, доктор фізико-математичних наук, професор П.Ф.Жук; директор комп'ютерного центру, інституту "Межрегіонального бізнесу", кандидат фізико-математичних наук, доцент М.С. Львов.

Методичні рекомендації
обговорено на засіданні кафедри
інформаційних технологій
Протокол № 11
від 6 червня 2000 р.

Схвалено навчально-
методичною радою
університету
Протокол №3
від 26.06.2000 р.

Рекомендовано до видання
Вченою радою Херсонського
державного педагогічного
університету
Протокол №7
від 3.07.2000 р.

Зміст

ВСТУП

В основі тієї або іншої мови програмування лежить деяка керівна ідея, що надає істотного впливу на стиль відповідних програм.

Об'єктно-орієнтоване програмування (ООП) є новим розумінням того, що власне називається обчисленнями, а також того, як ми можемо структурувати інформацію у середині комп'ютера. Це еволюційний крок, природним способом спливає із попередньої історії.

Історично першою з'явилася ідея процедурного структурування програм, відповідно до якої програміст повинний був вирішити, які саме процедури він буде використовувати у своїй програмі, а потім вибрати найкращі алгоритми для реалізації цих процедур. Поява цієї ідеї було наслідком недостатньої вивченості алгоритмічного боку обчислювальних процесів. Типовим прикладом процедурно-орієнтованої мови є Фортран.

В міру прогресу в галузі обчислювальної математики акцент у програмуванні почав зміщуватися з процедур убік організації даних, тому що ефективна розробка складних програм вимагає в діючих способах контролю правильності використання даних. Контроль повинний здійснюватися як на стадії компіляції, так і при прогоні програм, особливо при створенні великих програмних проектів. Виразне усвідомлення цієї проблеми привело до створення Алгола-60, а пізніше - Паскаля, Модула-2, Сі і множини інших мов програмування, що мають більш-менш розвинені структури типів даних. Логічним слідством розвитку цього напрямку став модульний підхід до розробки програм.

Починаючи з мови Симула-67, у програмуванні намітився новий підхід, що одержав назву об'єктно-орієнтованого програмування. Його керівна ідея полягає в прагненні зв'язати дані з процедурами, що обробляють ці дані в єдине ціле - об'єкт. Характерною рисою об'єктів є інкапсуляція (об'єднання) даних і алгоритмів їх опрацювання, у результаті чого і дані, і процедури багато в чому втрачають самостійне значення. Фактично об'єктно-орієнтоване програмування можна розглядати як модульне програмування нового рівня, коли при об'єднанні процедур і даних акцент робиться на їхній змістовний зв'язок.

Переваги ООП повною мірою виявляються лише при розробці досить складних програм. Більш того, інкапсуляція додає об'єктам цілком особливої властивості «самостійності», максимальної незалежності від інших частин програми. Правильно сконструйований об'єкт має у своєму розпорядженні всі необхідні дані і процедури їх опрацювання, щоб успішно реалізувати необхідні для нього дії. При розробці складних діалогових програм програміст змушений структурувати програму, тому що тільки в цьому випадку він може розраховувати на успіх: «критичною масою» неструктурованих програм є об'єм у 1000-1200 рядків вихідного тексту - налагодження неструктурованих програм більшого об'єму звичайно стикається з надмірними труднощами. Структурування програми веде, фактично, до розробки власної бібліотеки програмування – саме у цей момент до Вас на допомогу і приходять нові засоби ООП.

Об'єктно-орієнтоване програмування - це не просто декілька нових властивостей, доданих у вже існуючі мови. Скоріше - це новий крок в осмисленні процесів декомпозиції задач і розроблення програмного забезпечення.

ООП розглядає програми як сукупність вільно (гнучко) зв'язаних між собою агентів, що називані об'єктами. Кожний із них відповідає за конкретні задачі. Обчислення здійснюється за допомогою взаємодії об'єктів. Отже, у певному змісті програмування - це ні багато, ні мало, як моделювання миру.

Об'єкт одержується в результаті інкапсуляції стана (даних) і поведінки (операцій). Тим самим об'єкт у багатьох відношеннях аналогічний модулю або абстрактному типу даних.

Таким чином, курс "ООП" покликаний:

- сформувати у студентів уявлення про революційні ідеї програмування;
- роз'яснити фундаментальні концепції об'єктного підходу;
- допомогти опанувати термінологію і методи об'єктно-орієнтованого підходу;
- навчити застосовувати на практиці об'єктно-орієнтоване проектування, використовуючи одну з об'єктних мов;

Розроблені лабораторні роботи з цієї дисципліни:

1. знайомлять студентів з описом і принципами побудови складних систем;
2. дають уявлення про етапи створення складних систем, використовуючи об'єктний підхід: об'єктно-орієнтований аналіз, об'єктно-орієнтоване проектування, об'єктно-орієнтоване програмування;
3. формують у студентів навики опису програм і постановки задачі для формування навиків, необхідних для проведення факультативних занять з учнівськими;
4. сприяють формуванню у студентів абстрактного мислення, що повинно допомагати рішенню прикладних задач, зв'язаних з різними галузями знань.

Студенти повинні знати правила техніки безпеки роботи у комп'ютерному кабінеті.
Після виконання лабораторних робіт студенти повинні вміти:

- складати алгоритми, перекладати їх на мову Pascal;
- працювати у середовищі програмування Turbo Pascal 7.0;

проводити компіляцію, тестування, покрокове виконання, відладку програм.

Лабораторні роботи побудовані на принципах неперервності та системності навчання. Їх можна поділити на два блоки:

- модулі;
- об'єкти.

Для фізико-математичного та психолого-природничого факультетів передбачають:

1. Створення модуля CMPLX (модуль арифметики комплексних чисел)
2. Створення модуля Rac-ch (модуль арифметики раціональних чисел).
3. Розробка об'єктів (раціональне число, вектор, матриця векторів раціональних чисел).

Для факультету іноземної філології припускають:

1. Створення модуля введення даних
2. Створення модуля виведення даних.
3. Створення розрахунку модулю.
4. Розробка об'єктів по створенню бази даних по факультету.

Курс розрахований на студентів, що починають вивчати технології програмування. Оскільки концепції об'єктно-орієнтованого проектування припускають проектування не власне обчислювальних процесів, а опис об'єктів практично будь-якої предметної галузі спеціальні знання дисциплін математичного циклу не потрібно. На заняттях у якості прикладів для спеціальностей різних факультетів були підібрані об'єкти предметної галузі найбільше близької до спеціалізації студентів.

Курс "Об'єктно-орієнтоване програмування", для ефективного проведення лабораторних занять припускає знання студентами процедурної мови програмування (наприклад, Pascal), знання операційної системи Windows 95, уміння працювати з прикладними програмами.

Рекомендована література:

№п/п	Перелік підручників, навчальних та навчально-методичних посібників.
1.	Уолнєш К. Объектно-ориентированное программирование на языках BOR-LAND, C++ / Перевод с анг.; Худ. обл. М.В. Драко. – Мн. : ООО “Попурри”, 1990. – 640 с.: ил.
2.	Фаронов В.В. Турбо Паскаль 7.0. Начальный курс. Учебное пособие. – М.: “Нагидж”, 1997. – 616 с.: ил.
3.	Введение в программирование: / Автор-составитель В.А. Гольденберг. – Учебное пособие для учащихся среде. и стар. школн. возраста. – М.: ООО “Харвест”, 1997. – 528 с. (библиотека школьника)
4.	Прайс Д. Программирование на языках Паскаль: Практическое руководство. Пер. с англ. – М.: Мир, 1987. – 232 с.: ил.
5	Белецкий Я. Турбо Паскаль с графикой для персональных компьютеров/Пер. с польск. Д.И.Юренкова – М.:Машиностроение, 1991. – 320 с.
6	Фаронов В.В. Турбо Паскаль (в 3-х книгах). Кн. 2 Библиотека Turbo Vision. – М.: Учебно-инженерный центр «МВТУ – Фесто Дидактик», 1993. –412 с.
7	Тимоти Бадд Объектно-ориентированное программирование в действии – Пер. с англ. – СПб.: Питер, 1997.– 464 с.:ил.

Лабораторні роботи для студентів фізико-математичного та психолого-природничого факультетів.

Лабораторна робота №1
(6 годин)

Тема: Процедури та функції при програмуванні операцій та функцій з комплексними числами та змінними.

Ціль: Опанувати практичними навиками використання процедур та функцій для реалізації арифметики комплексних чисел у програмуванні.

Короткі теоретичні відомості

Алгебраїчна форма запису комплексного числа:

Z = A + jB = ReZ+ jImZ, де A - дійсна частина, а B - мнима частина комплексного числа Z, j - мнима одиниця.

Тригонометрична форма запису комплексного числа:

$Z = Re^{j\theta}$, де $R = \sqrt{A^2 + B^2}$, θ = arctg(B/A).

Завдання: Описати у вигляді процедур або функцій дані операції та функції з комплексними числами та змінними. Скласти програму , що демонструє роботу з розробленими процедурами та функціями.

Варіанти завдань:

1. Z_1+Z_2, де $Z_1=A_1+jB_1$, $Z_2=A_2+jB_2$
2. Z_1-Z_2
3. Z_1*Z_2
4. Z_1/Z_2
5. Z^2 , де Z=A+jB
6. 1/Z
7. $1/Z^2$
8. $\sqrt{z}$
9. exp Z
10. ln Z
11. $\sqrt[n]{z}$, де n>0
12. Z^n , де n>=0
13. sin Z
14. cos Z
15. tg Z

Лабораторна робота №2

(14 годин)

Тема: Модулі.

Ціль: Вивчити засоби мови ТР 7.0, за допомогою яких можна розбивати великі і складні програми на окремі логічно зв'язані частини, що створюються і компілюються незалежно один від одного і потім зв'язуються певним засобом. Оволодіти практичними навичками створення, оформлення і підготовки до роботи модуля в програмуванні.

Короткі теоретичні відомості

Модуль - це автономно компільована програмна одиниця.

Структура модуля. Модуль має наступну структуру:

```
unit <ім'я модуля>;
interface
<интерфейсна частина>
implementation
< реалізаційна частина >
[begin
< ініціалізаційна частина>]
end.
```

Заголовок модуля.

На відміну від заголовка програми, наявність заголовка модуля обов'язково. Ім'я модуля повинно збігатися з ім'ям файла, у якому модуль зберігається на диску.

Интерфейсна частина містить опис всіх програмних об'єктів модуля(типи, константи, змінні, процедури і функції), що повинні бути доступні основній програмі або іншим модулям.

Реалізаційна частина містить тіла глобальних процедур та функцій, а також локальні для модуля програмні об'єкти (мітки, константи, типи та ін.).

Ініціалізаційна частина може бути відсутня разом з відкриваючим її кодовим словом begin, бути порожньою, або вона може містити оператори, що виконуються до початку виконання основної програми.

Кодове слово end із точкою обов'язкове в кінці модуля.

Зв'язок і компіляція модуля. У результаті компіляції модуля створюється однойменний файл, але вже з розширенням **.TPU**, який поміщається в каталог, визначений опцією **Directories** підміню **Options** (EXE&TPU Directory).

Директива **Uses <список модулів>** підключає зазначені модулі до основної програми і їй стануть доступні об'єкти даних модулів.

Завдання 1. (4години) Оформити у вигляді модуля розроблені процедури та функції в лабораторній роботі №1. Скласти програму, що демонструє роботу із даним модулем.

Завдання 2. (10 годин) Створити модуль арифметики комплексних чисел, що об'єднує всі операції та функції наведені в лабораторній роботі №1 (дивись варіанти завдань). Скласти програму, що демонструє роботу, із даним модулем.

Примітка. При комплектації модуля арифметики комплексних чисел скористатися розробками студентів вашої підгрупи.

Лабораторна робота №3

(8 годин)

Тема: ООП. Основні властивості об'єктів. Інкапсуляція.

Ціль: Вивчити засоби мови ТР 7.0 які володіють одним із сучасних методів проектування програм, описаних як об'єктно-орієнтоване програмування (ООП). Опанувати потужними засобами ООП, що роблять програму більш структурірованою, легко підтримуваною і легко розширюваною. Розглянути один із основних властивостей об'єктів - інкапсуляція.

Короткі теоретичні відомості

Об'єкт - це структура даних, що містить поля даних(подібно запису) різних типів і заголовки методів.

Синтаксис об'єкта:

```
ИмяПотомка=
        object(ИмяПредка)
                поле;
                …
                поле;
                метод;
                …
                метод;
        end;
```

Метод - це процедура або функція, оголошені усередині оголошення елемента типу об'єкт.

Синтаксис: procedure Метод(Параметр1,Параметр2: integer);

Метод має доступ до полів даних об'єкта, не вимагаючи передачі їх йому у вигляді параметрів.

Тіло методу визначається поза оголошенням об'єкта. Його заголовок повинний містити ім'я об'єкта, якому належить метод. На приклад:

```
procedure ТипОб'єкта.Метод
            (Параметр1,Параметр2: integer);
Begin
    …
    …
end;{ Метод}
```

Методи підрозділяються на статичні і віртуальні. За допомогою віртуальних методів є можливість будувати ієрархію об'єктів з однаковими назвами методів, однак, різними кодами.

Синтаксис віртуального методу:

Procedure Метод(Параметр1,Параметр2:integer); virtual;

Крім звичайних процедур і функцій ТР 7.0 реалізує два спеціальних типи методів: конструктор і деструктор.

Конструктор - це спеціальний метод, що ініціює об'єкт , що містить віртуальні методи.

Синтаксис : constructor Init(Параметр1,Параметр2:integer);

Деструктор - це спеціальний метод, що звільняє пам'ять купи від динамічних об'єктів.

Синтаксис : destructor Done;

Основні властивості об'єктів.

Основними властивостями об'єкта є:

1. Інкапсуляція - об'єднання записів з процедурами та функціями, що працюють із цими записами.

Розглянемо зміст приведеної властивості.

Припустимо, що наші практичні інтереси лежать в області побудови зображень тіл зоряного неба в двохмірній площі. Основою всякого зображення є положення(позиція) окремого елемента на екрані, описаного координатами X і Y. Для того щоб задати двохмірну позицію підходить тип запису.

```
type pozition = record
                X,Y:integer;
end;
```

Що можна робити з парою координат (X,Y)?

По-перше, може знадобитися задати значення координат (у програмуванні така процедура зветься ініціалізація).

```
procedure Init(CoordX, CoordY:integer);
begin
      X:= CoordX;
      Y:= CoordY;
end;
```

По-друге, може знадобитися знання фактичних значень координат, для цього вводимо дві функції

```
function GetX:integer;
begin
        GetX:=X;
end;

function GetY:integer;
begin
       GetY:=Y;
end;
```

По нашому задуму процедура Init і функція GetX і GetY повинні працювати з полями запису Pozition.

Введення об'єктів у TP 7.0 дозволяє зафіксувати це положення, оголосивши поля і дії в одному місці.

```
type pozition = object
                X,Y:integer;
                procedure Init(CoordX,CoordY:integer);
```

```
        function GetX:integer;
        function GetY:integer;
end;
```

Процедура або функція, оголошені усередині об'єкта, називаються методами.

Тепер для ініціалізації екземпляра типу Pozition досить викликати його метод, як якби він був полем запису:

```
var  FirstPozition : Pozition;
…
FirstPozition.Init(10,15);
```

Метод задається так само, як і процедура в модулі: усередині об'єкта записується заголовок, при цьому всі поля, використовувані методом, повинні передувати його оголошенню. Визначення методу (розшифровування дій) відбувається поза оголошенням об'єкта, якому метод належить, супроводжуваним точкою.

```
procedure Pozition. Init(CoordX,CoordY:integer);
begin
      X:= CoordX;
      Y:= CoordY;
end;
```

Примітка. Імена формальних параметрів методу не можуть збігатися з іменами полів даних об'єкта.

Завдання. Перетворити модуль арифметики комплексних чисел (дивись завдання 2 лабораторна робота №2), представивши комплексне число , як об'єкт. Заінкапсуліруйте розроблені процедури та функції над комплексними числами і змінними . Скласти програму, що демонструє роботу, з даним модулем.

Лабораторна робота №4

(24 години)

Тема: ООП. Основні властивості об'єктів. Спадкування. Поліморфізм.

Ціль: Вивчити засоби мови ТР 7.0 які володіють одним із сучасних методів проектування програм, описаних як об'єктно-орієнтоване програмування (ООП). Опанувати потужними засобами ООП, що роблять програму більш структурірованою, легко підтримуваною і легко розширюваною. Розглянути основні властивості об'єктів - спадкування та поліморфізм..

Короткі теоретичні відомості

Об'єкт - це структура даних, що містить поля даних (подібно запису) різних типів і заголовки методів. Основні визначення типу "об'єкт" були розглянуті в розділі короткі теоретичні відомості лабораторної роботи №3.

Основні властивості об'єктів.

Основними властивостями об'єктів є:

1. Інкапсуляція - об'єднання записів із процедурами і функціями, що працюють з цими записами (зміст даної властивості розглянуто у лабораторній роботі №3).
2. Спадкування - завдання об'єкта, потім використання його для побудови ієрархії породжених об'єктів із спадкуванням доступу кожного з породжених об'єктів до коду і даним предка.
3. Поліморфізм - завдання одного імені дії, що передається нагору і вниз по ієрархії об'єктів, із реалізацією цієї дії засобом, що відповідає кожному об'єкту в ієрархії.

Розглянемо зміст 2-ї та 3-ї властивостей.

Спадкування.

Розглянемо зірку з координатами X і Y. Її можна зробити видимою або невидимою, їй можна задати колір, її можна перемістити.

Створимо об'єкт із такими можливостями

```
Star = object
        X,Y:integer;
        procedure Init(CoordX,CoordY:integer);
        function GetX:integer;
        function GetY:integer;
       Visible:boolean;
       Color:word;
        procedure Init(CoordX,CoordY:integer;InitColor:word);
        function IsVisible:boolean;
        procedure Show;{запалює зірку}
        procedure Blink;{гасить зірку}
        procedure Jump(NextX,NextY:integer);{переміщає зірку}
end;
```

Зауважимо, що поля X і Y і методи GetX, GetY практично збігаються з відповідними полями і методами об'єкта Pozition (дивись лабораторну роботу N3).

TP 7.0 надає можливість врахувати цю ситуацію. Варто вважати тип об'єкта Star породженим типом Pozition, записавши це в такий спосіб.

```
Star = object (Pozition)
        procedure Init(CoordX,CoordY:integer);
        function GetX:integer;
        function GetY:integer;
       Visible:boolean;
       Color:word;
        procedure Init(CoordX,CoordY:integer;InitColor:word);
        function IsVisible:boolean;
        procedure Show;{запалює зірку}
        procedure Blink;{гасить зірку}
        procedure Jump(Next,Next:integer);{переміщає зірку}
end;
```

Об'єкт Star тепер успадковує властивості об'єкта Pozition. Поля X,Y явно не задані в Star, але Star ними володіє завдяки спадкуванню, тобто можна написати

Star. X:=17;

Зміст об'єктно-орієнтованого програмування полягає в роботі з полями об'єкта через його методи.

Поліморфізм.

TP 7.0 дозволяє зберегти нащадку ім'я батьківського методу, "перекриваючи" його. Щоб перекрити батьківський метод, потрібно просто задати його з тим же ім'ям, але з другим тілом (кодом) і, якщо необхідно, з іншим набором параметрів. Такий метод робиться віртуальним і до його об'явлення добавляється слово virtual. Застосування віртуальних методів накладає обмеження на процедури ініціалізації, що повинні записуватися з зарезервованим словом constructor і мати загальне ім'я Init.

Кожний окремий екземпляр об'єкта повинний ініціалізуватися за допомогою окремого виклику конструктора.

Для очищення й знищення динамічно розподілених об'єктів існує спеціальна процедура **destructor Done**.

Завдання. Написати програму яка розраховує розмір стипендій та заробітної плати студентам, викладачам та співпрацівникам кафедри.

Описати типи об'єктів, які мають такі дані як прізвище, день виплати (отримання грошей), розмір ставки - для всіх категорій працівників кафедри, а також середній бал оцінок для студентів, розмір премії для співпрацівників та викладачів і кількість відпрацьованих годин та вартість години для викладачів.

Сума виплат визначається так:

для студентів: сума = ставка * середній бал;

для співпрацівників : сума = ставка + премія;

для викладачів: сума = ставка + премія + кількість годин * вартість години.

Передбачити можливість об'єктів обраховувати для себе суми виплат і загальний вивід значень даних об'єкта.

Розробити програму, що друкує розрахунковий лист кафедри.

Лабораторні роботи для студентів факультету іноземної філології.

Лабораторна робота № 1.

(8 години)

Основи модульного програмування.

Теоретичні відомості.

Модуль – це автономно компіліруєма програмна одиниця, яка включає до себе компоненти розділів описання (типи, константи, змінні, процедури та функції) , інколи, деякі оператори, що виконуються.

Модуль має наступну структуру:

```
UNIT  < ім'я > ;
INTERFACE
< інтерфейс на частина >
IMPLEMENTATION
< частина, що виконується >
                              BEGIN
< частина, що ініціїрується >
END.
```

Заголовок має зарезервіроване слово **UNIT** і ім'я модуля. Це ім'я повинно співпадати з іменем дискового файлу, в якому розміщується текст модуля.

Наприклад:

```
UNIT  Global;
```

текст модуля знаходиться у файлі – Global.pas

Модуль може використовувати інші модулі. Список модулів, з якими встановлюється зв'язок починається з слова **USES**, яке може бути розташовано після слів INTERFACE або IMPLEMENTATION.

Інтерфейсна частина використовується для об'яви усіх глобальних об'єктів модуля (типів, констант, змінних та підпрограм).

Наприклад:

```
UNIT  Global;
      INTERFACE
      type    complex = record
                           x , y : real
                        end ;
      procedure  add ( z : integer; var q : complex ) ;
```

Якщо у основній програмі об'явити *USES Global*, то стануть доступні процедура *add* та тип *complex*.

Частина, що виконується, має описання підпрограм, що були об'явлені у інтерфейсній частині. У цій частині модуля можна об'являти локальні для модуля об'єкти. У заголовку процедури або функції можна опускати описання формальних об'єктів, тому що вони вже описанні у інтерфейсній частині.

Приклад:

```
UNIT  Global;
    INTERFACE
    type    complex = record
                          x , y : real
                      end ;
    procedure  add ( z : integer; var q : complex ) ;
    IMPLEMENTATION
    procedure  add ;
        begin
            q.x:=5*z;
            q.y:=q.x+z;
        end;
        end.
```

Частина, що ініціїрується, завершує модуль. Вона може бути відсутньою (слова *begin* не має, залишається тільки слово *end*), або бути пустою – тоді за словом *begin* йде відразу слово *end.* У ініціїруючій частині розміщуються оператори, що виконуються, і які мають деякий фрагмент програми. Ці оператори виконуються до передачі управління основній програмі.

Приклад:

```
UNIT  Global;
    INTERFACE
    type    complex = record
                          x , y : real
                      end ;
    procedure  add ( z : integer; var q : complex ) ;
    IMPLEMENTATION
    var f: text;
    procedure  add ;
        begin
            q.x:=5*z;
            q.y:=q.x+z;
        end;
    BEGIN
        assiqn(f,'name');
        rewrite(f)
    END.
```

Завдання:

Задачу розв'язати трьома варіантами:

1. Відповідно варіанту завдання скласти програму без використання процедур та функцій.
2. Скласти програму того ж завдання з використанням процедур або функцій.
3. Скласти третій варіант програми з використанням модулів.

Варіанти завдання:

1. Описати процедуру вирізки зі слова і склейки слів. Складіть програму, яка з даного масиву слів вибирає ті, які мають непарну довжину і вирізає з них 2k-1 середніх букв
2. Описати процедуру читання слова з файлу INPUT. Описати процедуру вирізки закінчення слова. Скласти програму яка з даного файлу слів вибирає слова, що починаються з даного початку і друкує їх закінчення.
3. Описати процедуру читання слова з файлу INPUT. Описати процедуру порівняння двох слів. Скласти програму, яка друкує прізвища людей з даним іменем. Список прізвищ та імен міститься в файлі INPUT.
4. Описати процедуру вирізки зі слова і склейки слів. Складіть програму, яка з даного масиву слів вибирає ті, які мають парну довжину та поміняти ліву та праву частини слів.
5. Описати процедуру читання слова з файлу INPUT. Описати процедуру вирізки початку слова. Скласти програму яка з даного файлу слів вибирає слова, що закінчуються даним словом і друкує їх початок.
6. Описати процедуру обернення слова, порівняння слів. Скласти програму, яка в даному наборі слів з файлу INPUT шукає і друкує обернені слова.
7. Описати процедуру пошуку найбільшого та найменшого елементу масиву. За допомогою цієї процедури скласти програму, яка визначає чи співпадають найбільший і найменший елементи двох масивів A[1..n], B[1..n].
8. Описати процедури пошуку середнього арифметичного елементів цього масиву. За допомогою цієї процедури скласти програму яка знаходить середнє арифметичне в масиві A[1..99].
9. Описати процедуру обчислення НОД і НОК двох чисел. За допомогою цієї процедури складіть програму, яка обчислює НОД і НОК масиву натуральних чисел.
10. Описати процедуру перевірки розкладу натурального числа в суму двох квадратів. Скласти програму яка вибирає з даного масиву ті і тільки ті числа, які розкладуються в суму двох квадратів.
11. Описати процедуру перевірки простоти числа. Описати процедуру перевірки числа на розклад в суму виду 1+2a. Скласти програму, яка вибирає з даного масиву чисел ті, які задовольняють двом вимогам.
12. Описати функцію f(x) – кількість голосних букв у слові . Скласти програму, яка знаходить слова, що мають задану кількість голосних.
13. Описати функцію f(x, y) – кількість букв x у слові y. Скласти програму, яка знаходить слово, що містить найбільшу кількість входжень даної букви.
14. Описати функцію f(x, y), яка перевіряє, чи можливо переставивши букви в слові x отримати слово y. Скласти програму, яка після введення слова y знаходить слова x, які знаходяться в файлі INPUT .
15. Описати функцію f(x), яка знаходить чи входить цифра у слово. Скласти програму яка друкує слова, що містять цифри.
16. Описати функцію f(x) – перевірки чи є подвоєння букв у слові. Скласти програму, яка знаходить слова, що мають подвоєння букв.
17. Описати функцію f(x) – довжина слова x. Скласти програму пошуку слова, що має найменшу довжину.
18. Описати функцію f(x) – кількість різних простих дільників числа x. Скласти програму пошуку всіх чисел, що мають k різних простих дільників. Числа знаходяться в INPUT.
19. Описати функцію f(x) – дробова частина (x). Скласти програму пошуку числа з найбільшою дробовою частиною з даного набору чисел, що знаходяться в INPUT.

Лабораторна робота № 2.

(8 годин)

Основні принципи роботи з графічним модулем GRAPH.

Теоретичні відомості.

Характерна особливість систем програмування, реалізованих на персональних комп'ютерах - використання розширень мови засобами обробки графічної інформації. В системі програмування Turbo Pascal ці засоби зосереджені в модулі GRAPH.

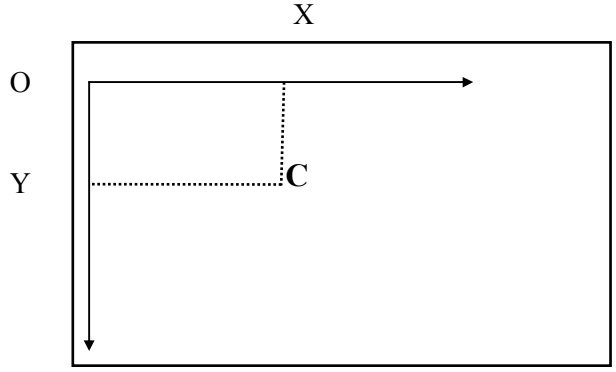
Модуль (UNIT) - це сукупність (бібліотека) описань констант, типів даних, змінних, процедур і функцій. Кожний модуль представляє собою по суті самостійну програму на Паскалі: він може містити декілька операторів, що виконуються перед викликом програми, і здійснюють всю необхідну ініціалізацію. Засоби, що описані в модулі, можна використовувати в програмі. Модуль можна компілювати окремо. Для того, щоб включити модулі в програму, необхідно описати їх у спеціальному розділі описань Uses. Синтаксичне описання цього розділу має вид:

Uses <список модулів >; {необов'язковий параметр}

Всі описання усередині модуля як правило пов'язані. Наприклад, стандартний зовнішній модуль CRT містить засоби, що підтримують взаємодію програми з екраном монітора в алфавітно-цифровому режимі (текстовий екран). Модуль GRAPH призначений для взаємодії програми з екраном у графічному режимі (графічний екран).

1. Графічний екран. Види графічних адаптерів.

Елементарною одиницею інформації, що виводиться в графічному режимі на екран, є точка, покрашена в деякий колір. Ця точка називається пікселом. Позиція пікселя на екрані визначається в декартовій системі координат (мал. 1). Відмітимо незвичний напрямок осі OY: вона направлена з лівого верхнього кута екрана униз.



Таким чином, кожний піксел описується трьома параметрами:

(X, Y, C), де C - колір пікселя.

X - ціле число з інтервалу [0 ... GetMaxX];

Y - ціле число з інтервалу [0 ... GetMaxY];

C - ціле число з інтервалу [0 ... GetMaxColors]

де GetMaxX, GetMaxY, GetMaxColors - константи, що описані в модулі GRAPH.

Діапазони допустимих значень X, Y, C визначаються технічними характеристиками апаратури комп'ютера, операційною системою і наладкою програми на один з можливих режимів використовування модуля GRAPH - ініціалізацією графіки. Детальні відомості про ініціалізацію графіки знаходяться в документації системи програмування.

Система програмування Turbo-Pascal в MS-DOS підтримує, зокрема, наступні графічні режими:

Відеоадаптер	GetMaxX	GetMaxY	GetMaxColor	Драйвер

CGA	**320**	**200**	**4 з 16**	**CGA.BGI**
MCGA	**320**	**200**	**16 з 256**	**CGA.BGI**
EGA	**640**	**200**	**16 з 256**	**EGAVGA.BGI**
EGA	**640**	**350**	**16 з 256**	**EGAVGA.BGI**
VGA	**640**	**480**	**16 з 256**	**EGAVGA.BGI**

Настройка програми на один з графічних режимів виконується автоматично оператором InitGraph, якщо відповідний драйвер установлений на комп'ютері.

InitGraph(< гр. адаптер >, < режим гр.др >, < путь >);

Перехід у текстовий режим здійснюється оператором CloseGraph.
Таким чином, програма, що використовує графіку, має вид:

```
Program MyGraph;
   Uses Graph;
    Var grDriver, grMode, errCode: Integer;
    .............
   Begin
    grDriver := Detect;
    InitGraph(grDriver, grMode, '< шлях до драйвера >'); { приклад шляху: D:\PasSys6\BGI }
    .............
   End.
```

Константа Detect визначена в модулі GRAPH. Її значення дорівнює 0. Detect вказує системі на необхідність підключення драйвера графіки, відповідного типу адаптера дисплея.

Графічний модуль GRAPH містить декілька десятків (більш 50) процедур і функцій, кожна з яких описана в документації системи і доступна програмісту з розділу HELP головного меню.

Тому ми приведемо тільки деякі відомості про GRAPH.

Процедури і функції:

Графічні примітиви:

PutPixel, GetPixel, GetX, GetY,
Line, LineTo, Move, Bar, DrawPoly, Rectangle, ...
Arc, Circle, Ellipse, ...

Обробка кольорів:

GetColor; GetPixel, GetBkColor, ...
SetColor, SetBkColor, SetPalette,...

Установка стиля малювання: SetLineStyle, SetFillStile, ...

Обробка тексту: OutText, OutTextXY, TextWidth, SetTextStyle, ...

Обробка графічного екрана: SetActivePage, PutImage, GetImage, SetViewPort,...

Процедура PutPixel(X, Y, C), наприклад, фарбує точку (X,Y) екрана в колір C. Одночасно графічний курсор поміщається в цю точку.

Процедура Line(X1,Y1, X2,Y2) малює відрізок AB кольором, що встановлений попередньо A = (X1, Y1), B = (X2, Y2).

Робота з текстами і введення-виведення в графічному екрані.

Введення-виведення інформації в режимі графіки суттєво відрізняються від відповідних процедур в алфавітно-цифровому режимі. Виведення тексту на екран здійснюють процедури OutText, OutTextXY.

OutText(TxtStr: String) виводить строку на пристрій виведення (графічний екран) в точку - позицію графічного курсора.

OutTextXY(X, Y: Integer; TxtStr: String) виводить строку на пристрій виведення (графічний екран) в точку (X, Y).

Крім цих процедур, в модулі GRAPH описані процедури і функції, підтримуючі різні стилі виведення. Так, наприклад:

Функції TextHeight(TxtStr:String), TextWidth(TxtStr:String) повертають відповідно висоту і ширину рядка в пікселах.

Процедура SetTextStyle(Font, Direction:Word; CharSize:Word) установлює стиль виведення тексту в графічному режимі, тобто визначає шрифт, стиль тексту і коефіцієнт збільшення символу. Як приклад опишемо процедуру WinText виведення надписів-коментарів у вікно графіка функції.

```
Procedure WinText;
    Begin
      SetColor(LightMagenta);
      SetTextStyle(0, 1, 1);
      OutTextXY(15,15, 'Function Graphics Window');
      SetColor(Cyan);
      SetTextStyle(0, 0, 1);
      OutTextXY(460,15, 'Production of SL.XXI');
      SetTextStyle(0, 0, 2);
      OutTextXY(28, 390, 'Y = Sqrt(Sin(|x|)');
     End;
```

Введення інформації в графічному режимі засобами модуля GRAPH не підтримується. Тому для організації введення необхідно скористуватися або стандартними процедурами, або процедурами модуля CRT (бібліотека алфавітно-цифрового режиму). На практиці це означає посимвольне введення інформації, формування рядка введених символів і перетворення типу рядка до типа даного, необхідного програмі.

Для введення символу використовується функція ReadKey - читання символу з клавіатури. Для відображення введеного символу на екрані (відлуння) використовується одна з графічних процедур виведення. Суперпозиція цих дій має вид Symbol := ReadKey; OutTextXY(x0, y0, Symbol). Для більш повної імітації процедури Read точка введення інформації на екрані звичайно відмічається зображенням курсора, який здвигається на одну позицію вправо при кожному введенні символу.

Завдання:

Намалювати і розфарбувати малюнок, згідно номеру варіанта.

Лабораторна робота №3

(8 годин)

Тема: Обєктно-орієнтоване програмування. Основні властивості об'єктів.

Короткі теоретичні відомості

Об'єкт - це структура даних, що містить поля даних (подібно запису) різних типів і заголовки методів.

Синтаксис об'єкта:

```
ИмяПотомка=
        object(ИмяПредка)
                поле;
                …
                поле;
                метод;
                …
                метод;
        end;
```

Метод - це процедура або функція, оголошені усередині оголошення елемента типу об'єкт.

Синтаксис:

```
procedure  Метод(Параметр1,Параметр2: integer);
```

Метод має доступ до полів даних об'єкта, не вимагаючи передачі їх йому у вигляді параметрів.

Тіло методу визначається поза оголошенням об'єкта. Його заголовок повинний містити ім'я об'єкта, якому належить метод. На приклад:

```
procedure ТипОб'єкта.Метод
            (Параметр1,Параметр2: integer);
Begin
    …
    …
end;{ Метод}
```

Методи підрозділяються на статичні і віртуальні. За допомогою віртуальних методів є можливість будувати ієрархію об'єктів з однаковими назвами методів, однак, різними кодами.

Синтаксис віртуального методу:

```
Procedure Метод(Параметр1,Параметр2:integer); virtual;
```

Крім звичайних процедур і функцій TP 7.0 реалізує два спеціальних типи методів: конструктор і деструктор.

Конструктор - це спеціальний метод, що ініціює об'єкт, що містить віртуальні методи.

Синтаксис :

```
constructor Init(Параметр1,Параметр2:integer);
```

Деструктор - це спеціальний метод, що звільняє пам'ять купи від динамічних об'єктів.

Синтаксис :

```
destructor Done;
```

Основні властивості об'єктів.

Основними властивостями об'єкта є:

1. Інкапсуляція - об'єднання записів з процедурами та функціями, що працюють із цими записами.
2. Спадкування - завдання об'єкта, потім використання його для побудови ієрархії породжених об'єктів із спадкуванням доступу кожного з породжених об'єктів до коду і даним предка.
3. Поліморфізм - завдання одного імені дії, що передається нагору і вниз по ієрархії об'єктів, із реалізацією цієї дії засобом, що відповідає кожному об'єкту в ієрархії.

Розглянемо зміст приведених властивостей.

Інкапсуляція.

Припустимо, що наші практичні інтереси лежать в області побудови зображень тіл зоряного неба в двохмірній площі. Основою всякого зображення є положення(позиція) окремого елемента на екрані, описаного координатами X і Y. Для того щоб задати двохмірну позицію підходить тип запису.

```
type pozition = record
                  X,Y:integer;
end;
```

Що можна робити з парою координат (X,Y)?

По-перше, може знадобитися задати значення координат (у програмуванні така процедура зветься ініціалізація).

```
procedure Init(CoordX, CoordY:integer);
begin
      X:= CoordX;
      Y:= CoordY;
end;
```

По-друге, може знадобитися знання фактичних значень координат, для цього вводимо дві функції

```
function GetX:integer;
begin
        GetX:=X;
end;

function GetY:integer;
begin
        GetY:=Y;
end;
```

По нашому задуму процедура Init і функція GetX і GetY повинні працювати з полями запису Pozition.

Введення об'єктів у TP 7.0 дозволяє зафіксувати це положення, оголосивши поля і дії в одному місці.

```
type pozition = object
                X,Y:integer;
                procedure Init(CoordX,CoordY:integer);
                function GetX:integer;
                function GetY:integer;
end;
```

Процедура або функція, оголошені усередині об'єкта, називаються методами.

Тепер для ініціалізації екземпляра типу Pozition досить викликати його метод, як якби він був полем запису:

```
var  FirstPozition : Pozition;
…
FirstPozition.Init(10,15);
```

Метод задається так само, як і процедура в модулі: усередині об'єкта записується заголовок, при цьому всі поля, використовувані методом, повинні передувати його оголошенню. Визначення методу (розшифровування дій) відбувається поза оголошенням об'єкта, якому метод належить, супроводжуваним точкою.

```
procedure Pozition. Init(CoordX,CoordY:integer);
begin
      X:= CoordX;
      Y:= CoordY;
end;
```

Примітка. Імена формальних параметрів методу не можуть збігатися з іменами полів даних об'єкта.

Спадкування.

Розглянемо зірку з координатами X і Y. Її можна зробити видимою або невидимою, їй можна задати колір, її можна перемістити.

Створимо об'єкт із такими можливостями

```
Star = object
        X,Y:integer;
        procedure Init(CoordX,CoordY:integer);
        function GetX:integer;
        function GetY:integer;
       Visible:boolean;
       Color:word;
        procedure Init(CoordX,CoordY:integer;InitColor:word);
        function IsVisible:boolean;
        procedure Show;{запалює зірку}
        procedure Blink;{гасить зірку}
        procedure Jump(NextX,NextY:integer);{переміщає зірку}
end;
```

Зауважимо, що поля X і Y і методи GetX, GetY практично збігаються з відповідними полями і методами об'єкта Pozition (дивись лабораторну роботу N3).

TP 7.0 надає можливість врахувати цю ситуацію. Варто вважати тип об'єкта Star породженим типом Pozition, записавши це в такий спосіб.

```
Star = object (Pozition)
        procedure Init(CoordX,CoordY:integer);
        function GetX:integer;
        function GetY:integer;
       Visible:boolean;
       Color:word;
        procedure Init(CoordX,CoordY:integer;InitColor:word);
        function IsVisible:boolean;
        procedure Show;{запалює зірку}
        procedure Blink;{гасить зірку}
        procedure Jump(Next,Next:integer);{переміщає зірку}
end;
```

Об'єкт Star тепер успадковує властивості об'єкта Pozition. Поля X,Y явно не задані в Star, але Star ними володіє завдяки спадкуванню, тобто можна написати

```
Star. X:=17;
```

Зміст об'єктно-орієнтованого програмування полягає в роботі з полями об'єкта через його методи.

Поліморфізм.

TP 7.0 дозволяє зберегти нащадку ім'я батьківського методу, "перекриваючи" його. Щоб перекрити батьківський метод, потрібно просто задати його з тим же ім'ям, але з другим тілом (кодом) і, якщо необхідно, з іншим набором параметрів. Такий метод робиться віртуальним і до його об'явлення добавляється слово virtual. Застосування віртуальних методів накладає обмеження на процедури ініціалізації, що повинні записуватися з зарезервованим словом constructor і мати загальне ім'я Init.

Кожний окремий екземпляр об'єкта повинний ініціалізуватися за допомогою окремого виклику конструктора.

Для очищення й знищення динамічно розподілених об'єктів існує спеціальна процедура **destructor Done.**

Завдання.

1. Описати об'єкти "планета", "зірка". Розробити програму, що демонструє рух планет сонячної системи.
2. Описати об'єкт "вікно". Розробити програму, що демонструє роботу користувача у віконному інтерфейсі.
3. Описати об'єкти "геометричні фігури" (точка, лінія, квадрат, коло). Розробити програму, що демонструє можливий рух фігур по екрану.
4. Описати об'єкт "студент". Розробити програму підрахунку студентських стипендій в залежності від середнього балу, отриманого при складанні екзаменів поточної сесії.
5. Описати об'єкти "шахові фігури". Розробити програму, що демонструє можливий рух шахових фігур по дошці.

Сторінки для зауважень студентів.

(Шановні студенти, велике прохання писати свої зауваження та запитання простим олівцем)