МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДОНЕЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

МЕТОДИЧНІ ВКАЗІВКИ ДЛЯ ВИКОНАННЯ КУРСОВОЇ РОБОТИ З ОБЧИСЛЮВАЛЬНОЇ МАТЕМАТИКИ ТА ПРОГРАМУВАННЯ

Донецьк ДонНТУ 2009

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДОНЕЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

МЕТОДИЧНІ ВКАЗІВКИ ДЛЯ ВИКОНАННЯ КУРСОВОЇ РОБОТИ З ОБЧИСЛЮВАЛЬНОЇ МАТЕМАТИКИ ТА ПРОГРАМУВАННЯ

(для студентів спеціальностей ХТ і ТХР напрямку 7.091604)

Розглянуто
на засіданні кафедри ВМ і П
протокол №10 від 22.05.09 р.

Затверджено на засіданні навчально-
видавничої ради ДонНТУ
протокол № 3 від 24.06.09г.
рег. № 203

Донецьк ДонНТУ 2009

УДК 681.3.06(071)

МЕТОДИЧНИЙ ПОСІБНИК ДЛЯ ВИКОНАННЯ КУРСОВОЇ РОБОТИ З ОБЧИСЛЮВАЛЬНОЇ МАТЕМАТИКИ ТА ПРОГРАМУВАННЯ / Укладачі: І.В. Диннік, О.А.Тихонова - Донецьк: ДонНТУ, 2009 - 54 с.

Методичний посібник містить завдання , теоретичний матеріал і методичні вказівки по оформленню курсової роботи з обчислювальної математики та програмування. Призначений для студентів спеціальностей ХТ і ТХР напрямку 7.091604.

Автори: И. В. Диннік, ст. викладач

О.А.Тихонова, асистент

Рецензент: І.Я.Зеленьова, доцент

Відповідальний за випуск
зав. кафедрою ОМіП В.М.Павлиш

ЗМІСТ

1. СТРУКТУРА ПОЯСНЮВАЛЬНОЇ ЗАПИСКИ

Пояснювальна записка повинна бути розділена на: вступну частину; основну частину; додаток (при необхідності).

Вступна частина містить наступні структурні елементи: титульну сторінку ; сторінку завдання і календарний план; реферат; зміст;

Основна частина повинна містити : вступ; суть проекту (роботи); висновки; перелік посилань.

Додатки розміщують після основної частини пояснювальної записки.

2.ВИМОГИ ДО СТРУКТУРНИХ ЕЛЕМЕНТІВ ОСНОВНОЇ ЧАСТИНИ

Титульна сторінка

Титульна сторінка є першою сторінкою пояснювальної записки і служить основним джерелом бібліографічної інформації, необхідної для обробки і пошуку документів.

Реферат

Реферат призначений для ознайомлення з роботою. Він повинний бути коротким, інформативним і містити інформацію, що дозволяє представити сутність роботи. Реферат повинний містити: інформацію про обсяг записки, кількість ілюстрацій, таблиць, додатків, кількість джерел по переліку посилань; текст реферату; перелік ключових слів.

Текст реферату повинний відображати інформацію, представлену в пояснювальній записці і, як правило, у визначеній послідовності: об'єкт дослідження; ціль роботи; методи дослідження; результати і їхня новизна; значимість роботи і висновки.

Реферат необхідно виконувати обсягом не більш 500 слів і розміщати на одній сторінці формату А4.

Ключові слова, істотні для розкриття суті записки, формують на основі тексту реферату і розміщують до тексту реферату. Перелік ключових слів включає від 5 до 15 слів (словосполучень), зображених (надрукованих) прописними літерами в називному відмінку в рядок через коми.

Зміст

Зміст розміщують безпосередньо після реферату, починаючи з нової сторінки. Зміст включає: вступ; послідовно перераховані найменування всіх розділів, підрозділів, пунктів і підпунктів (якщо вони маються); висновки; перелік посилань; найменування додатків.

Вступ

У вступі коротко викладають: оцінку сучасного стану проблеми; існуючі проблеми знання в даній області; ціль роботи. Вступ розташовують на окремій сторінці.

Перелік посилань

Перелік джерел, на які посилаються в записці, повинний бути приведений наприкінці тексту записки, починаючи з нової сторінки. У відповідних місцях записки повинні бути дані посилання. Бібліографічні описи в переліку посилань приводять у порядку, у якому вони вперше згадувалися в тексті. Порядкові номери описів у переліку є посиланнями в тексті (номерні посилання).

3.ПРАВИЛА ОФОРМЛЕННЯ ПОЯСНЮВАЛЬНОЇ ЗАПИСКИ

Загальні вимоги

Пояснювальну записку оформлюють на аркушах формату А4 (210х297) мм. Пояснювальну записку виконують рукописним, машинописним чи машинним (за допомогою комп'ютерної техніки) способом на одній стороні аркушу білого папера.

Текст записки варто друкувати, дотримуючи наступних розмірів полів: верхнє, ліве і нижнє - не менш 20 мм, праве - не менш 10 мм. Розмір шрифту-14 пт.Міжрядковий інтервал-полуторний.

У записці повинні бути чіткі, не розпливчасті лінії, букви, цифри й інші знаки. Усі букви, цифри і знаки повинні бути виконані чорним чорнилом по всій записці.

Помилки, описки і графічні неточності допускається виправляти чи підчищенням зафарбуванням білою фарбою і нанесенням на тім же чи місці між рядків виправленого зображення машинописним чи способом від руки.Кожен пункт, підпункт і перерахування записуються з абзацного відступу.

Розділи і підрозділи повинні мати заголовки. Пункти, як правило, заголовків не мають. Заголовки повинні чітко і коротко відбивати зміст розділів, підрозділв.

Заголовки структурних елементів записки і заголовки розділів варто розташовувати в середині рядка і друкувати прописними буквами без крапки наприкінці, не підкреслюючи. Заголовки підрозділів, пунктів і підпунктів записки варто починати з абзацного відступу і друкувати, крім першої, прописний малими літерами, не підкреслюючи, без крапки наприкінці. Якщо заголовок складається з двох речень, їх розділяють крапкою. Відстань між заголовком і текстом при виконанні записки повинна бути не менш двох рядків

Кожен розділ текстового документа варто починати з нової сторінки. Переноси слів у заголовку не допускаються.

Текст пояснювальної записки повинний бути коротким, чітким і не допускати різних тлумачень.

Структурні елементи «Реферат», «Вступ», «Висновки», «Перелік посилань» не нумерують, а їхні найменування служать заголовками структурних елементів. Розділи, підрозділи, пункти, підпункти варто нумерувати арабськими цифрами. Розділи записки повинні мати порядкову нумерацію в межах викладу суті,

наприклад 1, 2, 3, і т.д. Підрозділи повинні мати порядкову нумерацію в межах кожного розділу. Номер підрозділу складається з номера розділу і порядкового номера підрозділу, розділених крапкою. Після номера підрозділу крапку не ставлять, наприклад 1.1, 1.2 і т.д. Пункти повинні мати порядкову нумерацію в межах кожного розділу чи підрозділу, наприклад, 1.1, 1.2 чи 1.1.1, чи 1.1.2 і т.д.

Нумерація сторінок записки

Сторінки пояснювальної записки варто нумерувати арабськими цифрами, дотримуючи наскрізної нумерації по всьому тексту. Номер сторінки проставляють у правому верхньому куті сторінки без крапки наприкінці. «Титульну сторінку», «Завдання» включають у загальну нумерацію сторінок записки. Номера сторінок на титульній сторінці не проставляють. Ілюстрації і таблиці, розташовані на окремих сторінках, включають у загальну нумерацію сторінок записки.

Побудова таблиць

Таблиці застосовують для кращої наочності і зручності представлення показників. Назва таблиці, при її наявності, повинна відбивати її зміст, бути точною, короткою. Назва варто поміщати над таблицею. При переносі частини таблиці на ту ж чи інші сторінки назву поміщають тільки над першою частиною таблиці.

Таблиці, за винятком таблиць додатків, можна нумерувати арабськими цифрами наскрізною нумерацією у межах розділів. В останньому випадку номер таблиці складається з номера розділу і порядкового номера таблиці, відділених крапкою. При цьому перша цифра номер розділу, а друга – порядковий номер таблиці в розділі.

Таблиці кожного додатка позначають окремою нумерацією арабськими цифрами з додаванням перед цифрою позначення додатка. Якщо в документі одна таблиця, вона повинна бути позначена «Таблиця 1» чи «Таблиця В. 1 »,

якщо вона приведена в додатку В.

На всі таблиці повинні бути приведені посилання в тексті записки. При посиланні варто писати слово «таблиця (табл.)» із указівкою її номера. Наприклад, таблиця 2.1, що означає, що це перша таблиця другого розділу.

Заголовки граф і рядків таблиці варто писати з прописної літери, а підзаголовки граф – з малої літери, якщо вони складають одне речення з заголовком або із прописної літери, якщо вони мають самостійне значення. Наприкінці заголовків і підзаголовків таблиць крапки не ставляться. Заголовки і підзаголовки граф вказують в однині.

Розділяти заголовки і підзаголовки діагональними лініями не допускається. Горизонтальні і вертикальні лінії, що розмежовують рядки таблиці, допускається не проводити, якщо їхня відсутність не утрудняє користування таблицею. Заголовки граф, як правило, записують паралельно рядкам таблиці. При необхідності допускається перпендикулярне розташування заголовків граф.

Голівка таблиці повинна бути відділена від іншої частини таблиці. Висота рядків таблиці повинна бути не менш 8 мм. Таблицю, у залежності від її розміру, поміщають під текстом, у якому вперше дане посилання на неї, чи на наступній сторінці, а, при необхідності ,в додатку до документа.

Якщо рядки чи графи таблиці виходять за формат сторінки, таблицю поділяють на частині, поміщаючи одну частину під іншу, чи поруч, чи переносячи частину таблиці на наступну сторінку. При цьому в кожній частині таблиці повторюють її голівку і боковик.

При розподілі таблиці на частині допускається її голівку і боковик заміняти відповідно номером граф і рядків. При цьому нумерують арабськими цифрами графи і (чи) рядки першої частини таблиці.

Слово «Таблиця» вказують один раз ліворуч над першою частиною таблиці. Над іншими частинами пишуть слова «Продовження таблиці» із указівкою номера. Якщо наприкінці сторінки таблиця переривається і її

продовження буде на наступній сторінці, у першій частині таблиці нижню горизонтальну лінію, що обмежує таблицю, не проводять.

Графу «номер один по одному» у таблицю включати не допускається. При необхідності нумерації показників, параметрів чи інших даних порядкові номери варто вказувати в першій графі таблиці безпосередньо перед їх найменуванням. Перед числовими значеннями і позначенням типів, марок і т.п. порядкові номери в таблицях не проставляються.

Формули і рівняння

Формули і рівняння розташовують безпосередньо після тексту, у якому вони згадуються, в центрі сторінки. Вище і нижче кожної формули чи рівняння повинне бути залишено не менш одного вільного рядка.

Формули і рівняння в записці варто нумерувати порядковою нумерацією в межах розділу. Номер формули чи рівняння складається з номера формули чи рівняння, розділених крапкою, наприклад, формула (5.3) -третя формула п'ятого розділу. Номер формули чи рівняння вказують на рівні формули чи рівняння в дужках у крайньому правому положенні на рядку.

Пояснення значень символів і числових коефіцієнтів, що входять у формулу чи рівняння, варто приводити безпосередньо під формулою в тій послідовності, у якій вони дані в формулі чи рівнянні.

Пояснення кожного символу і числового коефіцієнта варто давати з нового рядка. Першу строку пояснень варто починати з абзацу словом «де» без двокрапки. Вище і нижче кожної формули чи рівняння повинне бути залишено не менш одного вільного рядка.

Додатка

Матеріал, що доповнює текст пояснювальної записки, допускається поміщати в додатках. Додатки повинні мати загальну з іншою частиною наскрізну нумерацію сторінок. У тексті пояснювальної записки на всі додатки

повинні бути дані посилання. Додатки розташовують у порядку посилань на них у тексті пояснювальної записки.

Кожен додаток варто починати з нової сторінки з указівкою нагорі в центрі сторінки слова «Додаток» і його позначення. Додаток повинний мати заголовок, що записують симетрично щодо тексту з прописної букви окремим рядком.

4.Завдання на курсову роботу

Тема курсової роботи : “Програмування з використанням підпрограм та типізованих файлів”.

Завдання на курсову роботу :

Розробити алгоритми визначених у варіанті підпрограм і основної програми, що використовує меню користувача, де кожен пункт меню викликає одну з підпрограм, і скласти основну програму. Окремими пунктами в меню має бути перегляд вихідних файлів, які будуються на основі приведених у варіанті завдання таблиць і файла, що містить результат . Розробити документацію до всіх структурних частин програми, в якій необхідно вказати призначення, вимоги до даних, що вводяться, формат інформації, що виводиться і дати опис розробленого алгоритму.

5. Варіанти завдань до курсової роботи

Варіант 1

Таблиця 1 (варіант 1)

Найменування товару	Одиниця виміру	Ціна 1 одиниці

Таблиця 2 (варіант 2)

Номер чека	Постачальник	Найменування товару	Кількість проданого товару	Номер відділу

Обчислити:

1. Вартість кожної покупки.
2. Суму податку по кожній покупці, що становить 20% від вартості.
3. Вартість кожної покупки за вирахуванням податку.
4. Виручку кожного відділу.
5. Виручку по кожному найменуванню товару.

Варіант 2

Таблиця 1 (варіант 2)

Тип касети	Ціна 1 хвилини запису за тарифом

Таблиця 2 (варіант 2)

Виконавець	Назва альбому	Час звучання (мін)	Кількість копій	Тип касети

Обчислити:

1. Вартість запису кожного альбому за тарифом.
2. Суму ПДВ, що становить 20% від вартості запису за тарифом.
3. Ціну альбому, включаючи ПДВ.
4. Загальні витрати на випуск альбому, враховуючи кількість копій.
5. Середню вартість записаного альбому за кожним типом касет.

Варіант 3

Таблиця 1 (варіант 3)

Аеропорт призначення	Ціна 1 квитка

Таблиця 2 (варіант 3)

Номер рейса	Аеропорт призначення	Дата вильоту	Місць в літаку	Продано квитків

Обчислити:

1. Вартість проданих квитків на кожен рейс.
2. Вартість непроданих квитків на кожен рейс.
3. Відсоток проданих квитків на кожен рейс.
4. Кількість вільних місць на кожен рейс.
5. Середню заповнюваність літаків по аеропортах призначення.

Варіант 4

Таблиця 1 (варіант 4)

Місто	Ціна 1 хвилини переговорів за тарифом

Таблиця 2 (варіант 4)

ПІБ абонента	Номер телефону	Місто	Дата переговорів	Кількість хвилин

Обчислити:

1. Вартість кожної розмови за тарифом.
2. Суму ПДВ, що становить 20% від вартості розмови за тарифом.

3. Вартість кожної розмови, включаючи ПДВ.
4. Суму до оплати для кожного абонента.
5. Середню тривалість розмови з кожним містом.

Варіант 5

Таблиця 1 (варіант 5)

Найменування приладу	Ціна приладу

Таблиця 2 (варіант 5)

ПІБ майстра	Номер цеху	Шифр приладу	Найменування приладу	Кількість приладів

Обчислити:

1. Вказати вартість приладів для кожного запису у відомості.
2. Кількість приладів, випущених кожним цехом.
3. Середня вартість приладів, випущених майстрами кожного цеху.
4. Вартість приладів, випущених кожним цехом.
5. Вартість випущених приладів кожного найменування.

Варіант 6

Таблиця 1 (варіант 6)

Найменування продукції	Одиниця виміру	Собівартість 1одиниці

Таблиця 2 (варіант 6)

Шифр продукції	Найменування продукції	Кількість продукції	Бригада

Обчислити:

1. Собівартість продукції, для кожного запису у відомості.

2. Суму ПДВ, що становить 20% від собівартості випущеної продукції.
3. Вартість випущеної продукції, включаючи ПДВ.
4. Середнє значення вартості, на яку випущено продукції кожною бригадою.
5. Суму, на яку випущено продукції кожного найменування.

Варіант 7

Таблиця 1 (варіант 7)

Найменування приладу	Час перевірки (мін)	Вартість перевірки

Таблиця 2 (варіант 7)

Код майстра	ПІБ майстра	Назва приладу	Кількість приладів

Обчислити:

1. Час поточної перевірки для кожного запису у відомості.
2. Суму, запрацьовану кожним майстром за поточну перевірку.
3. Загальну суму, запрацьовану кожним майстром.
4. Загальний час перевірки приладів кожним майстром окремо.
5. Загальний час перевірки приладів одного найменування.

Варіант 8

Таблиця 1 (варіант 8)

Назва деталі	Оплата за якісну деталь	Штраф за браковану деталь

Таблиця 2 (варіант 8)

Назва деталі	ПІБ робітника	Випущено (шт.)	Брак (шт.)

Обчислити:

1. Суму, запрацьовану кожним робітником по поточному запису у відомості. (браковані деталі не оплачуються).
2. Суму штрафу по кожному виду деталей.
3. Загальну суму до оплати по кожному робітникові.
4. Середнє значення оплати кожного робітника.
5. Загальна кількість якісних деталей кожного найменування.

Варіант 9

Таблиця 1 (варіант 9)

Назва виробу	Вартість випуску 1 виробу

Таблиця 2 (варіант 9)

Назва виробу	П І Б робітника	Номер цеху	Кількість виробів

Обчислити:

1. Вартість випуску кожного виду виробів кожним робітником.
2. Суму податку по різних виробах кожного робітника, що становить 20% від вартості випуску.
3. Вартість випущених кожним робітником виробів кожного виду, включаючи вартість випуску і податок.
4. Суму виробів, випущених кожним цехом.
5. Суму, на яку випущено виробів кожного найменування.

Варіант 10

Таблиця 1 (варіант 10)

Назва деталі	Витрати матеріалу (шт.)

Таблиця 2 (варіант 10)

Шифр деталі	Назва деталі	ПІБ робітника	Бригада	Випущено (шт.)	Брак (шт.)

Обчислити:

1. Витрати матеріалу на випуск кожного виду деталей кожним робітником.
2. Кількість матеріалу, витраченого на випуск бракованих виробів кожного виду кожним майстром.
3. Кількість матеріалу, витраченого на випуск якісних виробів кожного виду кожним робітником.
4. Витрати матеріалу на випуск деталей по кожній бригаді.
5. Загальна кількість якісних деталей кожного виду.

Варіант 11

Таблиця 1 (варіант 11)

Код книги	Вартість 1 екземпляра

Таблиця 2 (варіант 11)

Код книги	Назва книги	Автор	Номер магазина	Продано (шт.)

Обчислити:

1. Вартість книг кожного виду, проданих кожним магазином.
2. Суму націнки, що становить 10% від вартості по кожному виду книг, проданих кожним магазином.
3. Виручку по кожному виду книг, проданих кожним магазином, включаючи націнку.
4. Виручку кожного магазина.
5. Виручку по кожному автору.

Варіант 12

Таблиця 1 (варіант 12)

Код книги	Вартість 1 екземпляра

Таблиця 2 (варіант 12)

Код книги	Назва	Було (шт.)	Продано (шт.)	Номер магазина

Обчислити:

1. Виручку від продажу книг кожного коду, проданих кожним магазином.
2. Вартість непроданих книг кожного коду кожним магазином.
3. Кількість книг кожного кода, що залишилися в магазині.
4. Виручку від продажу книг по кожному магазину.
5. Суму виручки за кожним кодом книги.

Варіант 13

Таблиця 1 (варіант 13)

Артикул	Собівартість випуску 1 пари

Таблиця 2 (варіант 13)

Артикул	Різновид (жін., чол., дит.)	Сезон (літо, зима, весна-осінь)	Бригада	Випущено (шт.)

Обчислити:

1. Собівартість взуття кожного артикулу, випущеного кожною бригадою.
2. Суму націнки, що становить 25% від собівартості взуття кожного артикула, випущеною кожною бригадою.
3. Ціну взуття кожного артикулу, випущеного кожною бригадою, включаючи націнку.
4. Суму, на яку випущено взуття кожною бригадою.
5. Суму, на яку випущено взуття кожного артикулу.

Варіант 14

Таблиця 1 (варіант 14)

Артикул	Ціна 1 пари

Таблиця 2 (варіант 14)

Магазин	Артикул	Назва (туфлі, чоботи, кеди…)	Продано (шт.)

Обчислити:

1. Виручку від продажу кожного виду взуття кожним магазином
2. Суму податку, що становить 30% виручки від продажу кожного виду взуттю кожним магазином.
3. Суму, отриману кожним магазином за продаж кожного виду взуття за вирахуванням податку.
4. Виручку кожного магазина.
5. Виручку від продажу взуття кожного артикулу.

Варіант 15

Таблиця 1 (варіант 15)

Назва ліків	Ціна 1 упаковки

Таблиця 2 (варіант 15)

Аптека	Назва ліків	Місяць	Продано (упак.)

Обчислити:

1. Суму від продажу кожних ліків кожною аптекою.
2. Суму від продажу по кожному найменуванню ліків.
3. Суму від продажу по кожній аптеці.
4. Середню ціну 1 упаковки.
5. Загальна кількість всіх проданих упаковок.

Варіант 16

Таблиця 1 (варіант 16)

Пункт призначення	Ціна 1 квитка

Таблиця 2 (варіант 16)

Номер маршруту	Номер автобуса	Пункт призначення	Місць в автобусі	Продано

Обчислити:

1. Вартість проданих квитків на кожен автобус.
2. Кількість вільних місць в кожному автобусі.
3. Відсоток вільних місць в кожному автобусі.
4. Загальна кількість проданих квитків.
5. Середній відсоток вільних місць по кожному пункту призначення.

Варіант 17

Таблиця 1 (варіант 17)

Назва ліків	Ціна 1 упаковки

Таблиця 2 (варіант 17)

Номер аптеки	Назва ліків	Було (уп.)	Поступило (уп.)

Обчислити:

1. Кількість упаковок кожних ліків в кожній аптеці після надходження ліків.
2. Суму, на яку було кожних ліків в кожній аптеці до надходження ліків.
3. Суму, на яку поступило кожних ліків в кожну аптеку.
4. Суму, на яку поступило всіх ліків в кожну аптеку.
5. Суму, на яку поступило ліків кожного найменування.

Варіант 18

Таблиця 1 (варіант 18)

Назва ліків	Ціна 1 упаковки

Таблиця 2 (варіант 18)

Номер аптеки	Назва ліків	Група (від кашлю, сердцеві, …)	На початок дня (уп.)	Продано (уп.)

Обчислити:

1. Кількість упаковок кожних ліків в кожній аптеці на кінець дня.
2. Суму, на яку продано кожних ліків в кожній аптеці.
3. Суму, на яку було кожних ліків в кожній аптеці на початок дня.
4. Суму від продажу ліків по кожній аптеці.
5. Суму від продажу ліків по кожному найменуванню.

Варіант 19

Таблиця 1 (варіант 19)

Назва цукерок	Ціна 1 кг

Таблиця 2 (варіант 19)

Магазин	Назва цукерок	Різновид (карамель, шоколадні, …)	Продано (кг)

Обчислити:

1. Виручку кожного магазина по кожному найменуванню цукерок
2. Кількість цукерок, проданих кожним магазином.
3. Суму від продажу шоколадних цукерок всіма магазинами.
4. Виручку кожного магазина.

5. Виручку від продажу цукерок кожного найменування.

Варіант 20

Таблиця 1 (варіант 20)

Назва цукерок	Ціна 1 кг

Таблиця 2 (варіант 20)

Номер магазина	На початок дня (кг)	Продано (кг)	Отримано (кг)

Обчислити:

1. Кількість цукерок кожного найменування в кожному магазині в кінці дня
2. Суму, на яку продано цукерок кожного найменування кожним магазином.
3. Суму, на яку виявиться цукерок в кінці дня в кожному магазині.
4. Суму, на яку продано цукерок кожним магазином .
5. Суму, на яку продано цукерок кожного найменування.

Варіант 21

Таблиця 1 (варіант 21)

Крупа	Вартість 1 кг

Таблиця 2 (варіант 21)

Магазин	Крупа	На початок дня (кг)	Продано (кг)

Обчислити:

1. Вартість крупи кожного виду, проданої кожним магазином.
2. Суму націнки, що становить 10% від вартості крупи кожного виду, проданої кожним магазином.
3. Виручку від продажу кожним магазином крупи кожного виду, включаючи націнку.

4. Кількість крупи в кожному магазині на кінець дня.
5. Загальну виручку по кожному виду крупи.

Варіант 22

Таблиця 1 (варіант 22)

Назва крупи	Вартість за 1 кг

Таблиця 2 (варіант 22)

Номер магазина	Назва крупи	Продано (кг)	Залишилося (кг)

Обчислити:

1. Вартість крупи кожного найменування, проданої кожним магазином.
2. Вартість крупи кожного найменування, не проданої кожним магазином.
3. Кількість крупи кожного найменування, що є в наявності на початок дня в кожному магазині.
4. Суму, отриману від продажу всієї крупи кожним магазином.
5. Суму, отриману від продажу всієї крупи кожного найменування.

Варіант 23

Таблиця 1 (варіант 23)

Розряд	Вартість 1 години (грн.)

Таблиця 2 (варіант 23)

Табельний номер	ПІБ робітника	Номер цеха	Розряд	Відпрацьовано годин

Обчислити:

1. Заробіток робітників по кожному рядку відомості.

2. Суму податку, що становить 20% від зароблені суми.
3. Суму до виплати для кожного робітника за вирахуванням податку.
4. Суму, запрацьовану робітниками кожного цеху.
5. Суму, запрацьовану робітниками кожного розряду.

Варіант 24

Таблиця 1 (варіант 24)

Посада	Оплата за 1 день

Таблиця 2 (варіант 24)

ПІБ співробітника	Посада	Номер відділу	Відпрацьовано (днів)	Запізнень

Обчислити:

1. Суму, запрацьовану кожним співробітником за відпрацьовані дні.
2. Суму штрафу по кожному співробітникові, що становить 10% від оплати за 1 день запізнень.
3. Суму до виплати для кожного робітника за вирахуванням штрафу за запізнення.
4. Суму до виплати по кожному відділу.
5. Суму, запрацьовану співробітниками по кожній посаді.

Варіант 25

Таблиця 1 (варіант 25)

Вид тканини	Собівартість 1 $м^2$

Таблиця 2 (варіант 25)

Вид тканини	Ширіна (м)	Довжина (м)	Колір	Номер цеху

Обчислити:

1. Кількість (площа) тканини кожного виду, випущеної кожним цехом.

2. Собівартість тканини кожного виду, випущеної кожним цехом.
3. Націнку, що становить 15% від собівартості тканини кожного виду, пущеної для вас кожним цехом.
4. Кількість тканини, випущеної кожним цехом.
5. Кількість випущеної тканини по кожному виду.

Варіант 26

Таблиця 1 (варіант 26)

Артикул	Вартість 1 м

Таблиця 2 (варіант 26)

Артикул	Колір	Довжина (м)	Магазин

Обчислити:

1. Вартість тканини кожного артикулу, проданої кожним магазином.
2. Націнку, що становить 10% від вартості тканини кожного артикулу, проданої кожним магазином.
3. Суму від продажу кожним магазином тканини кожного артикулу, включаючи націнку.
4. Суму від продажу тканини по кожному магазину.
5. Суму від продажу тканини кожного артикулу

Варіант 27

Таблиця 1 (варіант 27)

Вид складності	Оплата 1 сторінки

Таблиця 2 (варіант 27)

Табельний №	П І Б друкарки	Кількість сторінок	Вид складності

Обчислити:

1. Суму, запрацьовану кожною друкаркою за роботу кожної складності.

2. Податок, що становить 22% від суми, запрацьованої кожною друкаркою за роботу кожної складності.
3. Суму до оплати кожній друкарці за роботу кожної складності за вирахуванням податку.
4. Суму до оплати кожній друкарці.
5. Суму до оплати по кожному виду складності.

Варіант 28

Таблиця 1 (варіант 28)

Вид складності	Оплата 1 сторінки

Таблиця 2 (варіант 28)

П І Б друкарки	Номер машбюро	Вид складності	Кількість сторінок

Обчислити:

1. Суму, запрацьовану кожною друкаркою за роботу кожної складності.
2. Кількість сторінок, виконаних кожною друкаркою.
3. Кількість сторінок, виконаних кожним машбюро.
4. Суму, запрацьовану кожною друкаркою.
5. Кількість виконаних сторінок по кожному виду складності.

Варіант 29

Таблиця 1 (варіант 29)

Артикул	Ціна 1 рулону

Таблиця 2 (варіант 29)

Магазин	Артикул	Колір	На початок дня (рул.)	Продано (рул.)

Обчислити:

1. Суму від продажу кожним магазином шпалер кожного артикулу.
2. Загальна кількість рулонів шпалер, проданих кожним магазином.
3. Відсоток проданих кожним магазином шпалер кожного артикулу від тих, що є на початок дня.
4. Суму від продажу шпалер по кожному магазину.
5. Суму від продажу шпалер кожного артикулу.

Варіант 30

Таблиця 1 (варіант 30)

Код банку	Позиковий відсоток

Таблиця 2 (варіант 30)

Місто	Код банку	Назва банку	Назва фірми	Видані позики (тис. грн.)

Обчислити:

1. Суму відсотків, отриманих кожним банком по кожній позиці.
2. Суму, яку кожен банк повинен отримати назад (включаючи проценти).
3. Суму виданих позик по кожному місту.
4. Суму виданих позик по кожному банку.
5. Суму, що отримується кожним банком у вигляді відсотків.

6.Теоретичний матеріал ,необхідний для виконання курсової роботи

6.1. Технологія програмування

Інтерес до інженерних питань програмування, до деталей процесу виготовлення програмного продукту, до формування промислової культури програмування обумовлений трудомісткістю й великою вартістю програмного

продукту, що може в кілька разів перевершувати вартість ЕОМ. Причиною такого росту витрат на виготовлення програмного продукту є те, що програмування складалося як творча діяльність, яка важко дисциплінується, і тому офіційно була названа мистецтвом. Кожен програміст пише програми, не дотримуючись яких-небудь чітких стандартів і правил. Стало майже неможливо розібратися до кінця, що робить програма, написана іншою людиною, і практично неможливо її налагодити або модифікувати. Виникла ситуація, коли стало простіше переписати програму заново, чим трохи її змінити. Надії на те, що нові універсальні мови програмування в корені змінять ситуацію, не виправдовуються. Тому виникла необхідність розробити деякі принципи, яких варто дотримуватися при написанні програм.

Метою програмування є опис процесів обробки даних (надалі - просто процесів). Під даними розуміється подання фактів й ідей у формалізованому виді, придатному для передачі й переробці в якомусь процесі. Обробка даних - це виконання систематичної послідовності дій з даними. Дані представляються й зберігаються на т.зв. носіях даних. Сукупність носіїв даних, використовуваних при якій-небудь обробці даних, прийнято називати інформаційним середовищем. Набір даних, що втримуються в який-небудь момент в інформаційному середовищі можна називати станом цього інформаційного середовища. Процес можна визначити як послідовність змінюючих одне одного станів деякого інформаційного середовища. Описати процес - значить визначити послідовність станів заданого інформаційного середовища. Формалізований опис процесу називається програмою. Програма складається формалізованою мовою програмування. Перш ніж скласти програму, доводиться проробляти більшу підготовчу роботу з уточнення постановки завдання, вибору методу її рішення, з'ясуванню специфіки застосування необхідної програми, проясненню загальної організації розроблювальної програми й багато чого іншого. Використання цієї інформації може істотно спростити завдання розуміння програми, тому досить

корисно її якось фіксувати у вигляді окремих документів (часто не формалізованих, розрахованих тільки для сприйняття людиною). Звичайно програми розробляються розраховуючи на те, щоб ними могли користуватися люди, що не беруть участь у їхній розробці (їх називають користувачами). Для освоєння програми користувачем крім її тексту потрібно певна додаткова документація. Програмна документація дозволяє зрозуміти, які функції виконує та або інша програма, як потрібно підготувати вихідні дані й запустити необхідну програму в процес її виконання, а також інтерпретувати одержувані результати. Крім того, програмна документація допомагає розібратися в самій програмі, що необхідно, наприклад, при її модифікації.

Під технологією програмування розуміється вдосконалювання професійної культури програмування, організації й упорядкування праці програміста, незалежно від конкретної мови програмування й розв'язуваного завдання.

Якими ж якостями володіє гарна програма?

- ✓ Вона працює.
- ✓ Вона працює відповідно до специфікації .
- ✓ Вона гнучка.
- ✓ Вона зроблена в строк.
- ✓ У ній немає помилок. (Помилки, які неминучі, швидко можуть бути виявлені й локалізовані).
- ✓ Вона добре оформлена.
- ✓ Вона має дружній інтерфейс.

Дуже важливе правило: "Програми пишуться для людей, а не для комп'ютера". Яку би складну й витончену програму Ви не написали, у випадку, якщо вона зрозуміла тільки Вам, якщо в ній відсутні коментарі, якщо вона погано оформлена, те ця програма марна.

Не слід плутати технологію програмування з методологією програмування. Хоча в обох випадках вивчаються методи, але в технології програмування

методи розглядаються "зверху" - з погляду організації технологічних процесів, а в методології програмування методи розглядаються "знизу" - з погляду основ їхньої побудови. Методологія програмування визначається як сукупність механізмів, застосовуваних у процесі розробки програмного забезпечення й об'єднаних одним загальним філософським підходом.

Одна з основних ідей, покладених у більшість відомих технологій програмування - спадне проектування. Існують також інші назви: "програмування з покроковим удосконалюванням", "систематичне програмування", "ієрархічне програмування". Принцип його - спочатку визначаються основні функції, які повинні бути забезпечені програмою, а потім остаточно визначаються додаткові функції, що випливають із основних функцій.

Наприклад:

Основна функція: " обробка файлу" .

Додаткові функції: " відкрити файл" , " обробити всі записи" , " закрити файл" .

6.2. Структурне програмування

Поняття структурного програмування в значній мірі визначаються властивостями мов програмування. Для написання програми з гарною структурою необхідно, щоб мова програмування, на якій написана програма, мала можливості добре структурованого опису процедур обробки. Одне із властивостей структурного програмування полягає в тому, що воно пропонує безліч різних способів, як написати програму для рішення якогось завдання. Сукупність всіх цих способів називається методологією програмування. Тут дивним образом проявляється процес перетворення програмування з мистецтва в науку.

Серед найважливіших понять методології програмування особливе значення мають ієрархічність (тобто багаторівневість) і приховання інформації,

засновані на відомому здавна принципі <розділяй і пануй>. Ще в імперії Чінгісхана для керування більшою державою довелося розділити його на трохи більше дрібних держав, і на чолі кожного із цих держав був поставлений свій правитель. Принцип <розділяй і пануй> виявився й у древні століття найефективнішим. Ідея багаторівневості, або ієрархічности, складається в створенні такої структури, у якій складові її частини побудовані приблизно по однаковому зразку. У Японії є іграшка, схожа на російську мотрійку, що являє собою вкладений друг у друга кілька шухлядок. Це і є типовий приклад простої ієрархічної структури.

Структура гарної програми, складеної з урахуванням вимог структурного програмування, у принципі повинна бути досить простій й являти собою сукупність блоків, не залежних по відношенню друг до друга або перебувають в ієрархічному зв'язку. Таким чином, уже на стадії програмування важливо врахувати майбутню ієрархічну структуру з дотриманням обох принципів структурування: ієрархічність по вертикалі й незалежність друг від друга елементів одного рівня по горизонталі. Отже, при складанні програми намагайтеся додержуватися принципу, про який ми тільки що говорили,- від цілого до частини з урахуванням ієрархії.

6.3.Модульне програмування.

Модульне програмування - це мистецтво розбивки завдання на деяке число різних модулів, уміння широко використати стандартні модулі.

Основні концепції модульного програмування:

- кожен модуль реалізує єдину незалежну функцію;
- кожен модуль має єдину точку входу й виходу;
- розмір модуля по можливості повинен бути мінімізований;
- кожен модуль може бути окремо протестований;
- вся програма побудована з модулів;

- модуль не повинен давати побічних ефектів;
- кожен модуль не залежить від того, як реалізовані інші модулі.

При такому підході складна система поділяється на кілька частин. Кожен модуль реалізує єдину функцію. Розмір модуля невеликий, тому тестування кероване й може бути проведено ретельним образом. Після кодування й тестування всіх модулів відбувається їхня інтеграція, і тестується вся програма. Практично у всіх мовах програмування є засоби структурування. Мови, у яких передбачені такі механізми, називаються процедурно- орієнтованими. До їхнього числа належить і Турбо Паскаль.

Процедури й функції являють собою важливий інструмент Турбо Паскаля, що дозволяє писати добре структуровані програми. У структурованих програмах звичайно легко прослідкується основний алгоритм, їх неважко зрозуміти, вони простіше в налагодженні й менш чутливі до помилок програмування. Всі ці властивості є наслідком важливої особливості підпрограм, кожна з яких являє собою багато в чому самостійний фрагмент програми, пов'язаний з основною програмою лише за допомогою декількох параметрів. Самостійність підпрограм дозволяє локалізувати в них всі деталі програмної реалізації тієї або іншої алгоритмічної дії й тому зміна цих деталей, наприклад, у процесі налагодження, звичайно не приводить до змін основної програми.

Процедурою в Турбо Паскалі називається особливим чином оформлений фрагмент програми, що має власне ім'я. Згадування цього імені в тексті програми приводить до активізації процедури й називається її викликом. Відразу після активізації процедури починають виконуватися оператори, що входять до неї, після виконання останнього з них керування повертається назад в основну програму й виконується оператор, що коштує безпосередньо за оператором виклику процедури.

Для обміну інформацією між основною програмою й процедурою використовується один або кілька параметрів виклику. Процедури можуть мати й

інший механізм обміну даними із визивною програмою, так що параметри виклику можуть і не використовуватися. Якщо параметри є, то вони перелічуються в круглих скобках за іменем процедури й разом з ним утворять оператор виклику процедури.

Функція відрізняється від процедури тим, що результат її роботи повертається у вигляді значення цієї функції, і, отже, виклик функції може використовуватися поряд з іншими операндами у виразах.

У Турбо Паскалі є багато стандартних процедур і функцій. Наявність багатої бібліотеки таких програмних заготівок істотно полегшує розробку прикладних програм. Однак іноді програмістові доводиться розробляти свої, нестандартні процедури й функції.

Нестандартні процедури й функції необхідно описати, щоб компілятор міг встановити зв'язок між оператором виклику й тих дій, які передбачені в підпрограмі. Опис підпрограми поміщається в розділі описів і зовні виглядає як програма, але замість заголовка програми фігурує заголовок процедури (функції).

Програмуючи з використанням підпрограм, ми знайомимось з універсальним методом конструювання складних програм, що отримали назву спадне програмування. Відповідно до цього методу створення програми починається «зверху», тобто з розробки самого головного, генерального алгоритму. На верхньому рівні звичайно ще не ясні деталі реалізації тієї або іншої частини програми, тому ці частини варто замінити тимчасовими заглушками. Бажано, щоб тимчасовий варіант програми був синтаксично правильним, тоді можна його відкомпілювати й переконатися у відсутності в ньому синтаксичних помилок. Такий прогін дасть визначену впевненість перед розробкою й реалізацією алгоритмів нижнього рівня, тобто перед заміною заглушок реально працюючими процедурами. Якщо реалізований у заглушці алгоритм досить складний, його знову структурують, виділяючи головний

алгоритм і застосовуючи нові заглушки, і т.д. Процес триває «униз» доти, поки не буде створений повністю працездатний варіант програми.

Опис підпрограми складається із заголовка й тіла підпрограми.

Заголовок процедури має вигляд:

PROCEDURE <ім'я> [(<сп. ф.з. >)] ;

Заголовок функції:

FUNCTION <ім'я> [(<сп.ф.з.>)] : <тип>;

Тут <ім'я> - ім'я підпрограми (правильний ідентифікатор);

<сп.ф. з. > - список формальних параметрів;

<тип> - тип результату, що повертає функцією.

Список формальних параметрів необов'язковий і може бути відсутнім. Якщо ж він є, то в ньому повинні бути перераховані імена формальних параметрів й їхні типи, наприклад:

Procedure SB(a: Real; b: Integer; c: Char);

Як видно із приклада, параметри в списку відокремлюються друг від друга крапками з комою. Кілька слідуючих один за одним однотипних параметрів можна поєднувати в підсписки, наприклад Procedure F(a , b: Real);

Оператори тіла підпрограми розглядають список формальних параметрів як своєрідне розширення розділу описів: всі змінні із цього списку можуть використовуватися в будь-яких виразах усередині підпрограми. Механізм заміни формальних параметрів на фактичні дозволяє потрібним способом настроїти алгоритм, реалізований у підпрограмі. Турбо Паскаль стежить за тим, щоб кількість і тип формальних параметрів строго відповідали кількості й типам фактичних параметрів у момент звертання до підпрограми. Значення використовуваних фактичних параметрів залежить від того, у якому порядку вони перераховані при виклику підпрограми. Користувач повинен сам стежити за правильним порядком перерахування фактичних параметрів при звертанні до підпрограми.

Любий з формальних параметрів підпрограми може бути або параметром-значенням, або параметр-змінною, або параметром-константою. Якщо параметри визначаються як параметри-змінні, перед ними необхідно ставити зарезервоване слово VAR, а якщо це параметра-константи - слово CONST, наприклад:

Procedure MyProcedure (var a: Real; b: Real; const c: String);

Тут a - параметр-змінна, b -параметр-значення, а c - параметр-константа.

Визначення формального параметра тим або іншому способом істотно тільки для визивної програми: якщо формальний параметр оголошений як параметр-змінна, то при виклику підпрограми йому повинен відповідати фактичний параметр у вигляді змінної потрібного типу; якщо формальний параметр оголошений як параметр-значення або параметр-константа, то при виклику йому може відповідати довільне вираження. Контроль за дотриманням цього правила здійснюється компілятором Турбо Паскаля.

Якщо параметр визначений як параметр-значення, то перед викликом підпрограми це значення обчислюється, отриманий результат копіюється в тимчасову пам'ять і передається підпрограмі. Важливо врахувати, що навіть якщо фактичний параметр зазначений як найпростіший вираз у вигляді змінної або константи, однаково підпрограмі буде передана лише копія змінної (константи). Будь-які можливі зміни в підпрограмі параметра-значення ніяк не сприймаються визивною програмою, тому що в цьому випадку змінюється копія фактичного параметра.

Якщо параметр визначений як параметр-змінна, то при виклику підпрограми передається сама змінна, а не її копія (фактично в цьому випадку підпрограмі передається адреса змінної). Зміна параметра-змінної приводить до зміни самого фактичного параметра в визивній програмі.

У випадку параметра-константи в підпрограму також передається адреса області пам'яті, у якій розташовується змінна або обчислене значення. Однак

компілятор блокує будь-які присвоювання параметру-константі нового значення в тілі підпрограми.

Отже, параметри-змінні використовуються як засіб зв'язку алгоритму, реалізованого в підпрограмі, із зовнішнім світом: за допомогою цих параметрів підпрограма може передавати результати своєї роботи визивній програмі. У розпорядженні програміста завжди є й інший спосіб передачі результатів - через глобальні змінні. Однак зловживання глобальними зв'язками робить програму , як правило, заплутаною, важкою в розумінні й складною в налагодженні. Відповідно до вимог гарного стилю програмування ,рекомендується там, де це можливо, використовувати передачу результатів через фактичні параметр-змінні.

Опис всіх формальних параметрів як параметрів-змінних небажано по двох причинах. По-перше, це виключає можливість виклику підпрограми з фактичними параметрами у вигляді виразів, що робить програму менш компактною. По-друге, і головне, у підпрограмі можливо випадкове використання формального параметра, наприклад, для тимчасового зберігання проміжного результату, тобто завжди існує небезпека ненавмисно зіпсувати фактичну змінну. От чому параметрами-змінними варто оголошувати тільки ті, через які підпрограма в дійсності передає результати визивній програмі. Чим менше параметрів оголошено параметрами-змінними й чим менше в підпрограмі використається глобальних змінних, тим менше небезпека одержання непередбачуваних програмістом побічних ефектів.

Існує ще одна обставина, яку варто враховувати при виборі виду формальних параметрів. При оголошенні параметра-значення здійснюється копіювання фактичного параметра в тимчасову пам'ять. Якщо цим параметром буде масив великої розмірності, то істотні витрати часу й пам'яті на копіювання при багаторазових звертаннях до підпрограми можна мінімізувати, оголосивши цей параметр параметром-константою. Параметр-константа не копіюється в

тимчасову область пам'яті, що скорочує витрати часу на виклик підпрограми, однак будь-які його зміни в тілі підпрограми неможливі - за цим строго стежить компілятор.

Оголошення змінних у списку формальних параметрів підпрограми відрізняється від оголошення їх у розділі опису змінних : типом будь-якого параметра в списку формальних параметрів може бути тільки стандартний або раніше оголошений тип. Тому не можна, наприклад, оголосити наступну процедуру:

Procedure S (a: array [1..10] of Real);

Якщо в підпрограму передається весь масив, то варто спочатку описати його тип. Наприклад:

type

atype = array [1..10]of Real;

Procedure S (a: atype);

Оскільки рядок є фактично своєрідним масивом, її передача в підпрограму здійснюється аналогічним способом:

type

intype = String [15] ;

outype = String [30] ;

Function St (s : intype): outype;

6.4.Стиль програмування. KISS-принцип

Перекладання хоча б частини витрат з голови людини на методику програмування іноді різко підвищує ефективність роботи мозку. Напевно, важко дати формальне визначення стилю програмування, однак, як у звичайному житті ми легко впізнаємо людину, що дотримується певного стилю поведінки, стилю одягу й так далі, так і, дивлячись на програму, легко визначити, чи дотримувався при її створенні програміст яких-небудь правил, традицій, або робив все, як

заманеться.

Безсумнівно, що дотримання якому-небудь набору правил у програмуванні не гарантує від створення поганого програмного продукту, але, з іншого боку, немає, і не може бути якісної програми, написаної без дотримання певного набору правил оформлення, документування, організації взаємодії з користувачем і тому подібного. Розглянемо деякі принципи, дотримуючись яких можна виробити гарний стиль програмування.

Відправною точкою у виборі стилю програмування є KISS-принцип : чим простіше програма, тим більше ймовірність того, що вона буде добре працювати. Починайте спрощення зі структури даних, інтерфейсу з користувачем, структури програми.

Можна виділити кілька правил, дотримання яких говорить про гарний стиль програмування.

Програма, написана в гарному стилі програмування, має імена змінних, констант, функцій, модулів, які несуть смислове навантаження, а не є безглуздим набором символів типу XXYY123. Ретельний вибір імен програмних об'єктів, робить програму зрозумілішою й допомагає позбутися помилок. Одним з підходів до вибору імен змінних, констант і тому подібних, є так називана "Угорська нотація". При цьому всі імена починаються з позначення типу елемента, далі йде власне його ім'я. Яскравим прикладом мови, стиль програмування якою "зав'язаний" на "Угорській нотації", є Borland DELPHI.

Використання відступів у вихідному тексті програми також зменшує кількість помилок і свідчить про гарний стиль. Як правило, відступами позначаються операторні дужки. Відступи роблять програму кращою для читання і легшою для модифікації.

6.5. Тестування програми

Епіграф : Нема чого на дзеркало нарікати...

Під налагодженням програми розуміється процес, що дозволяє отримати програму, що правильно працює з характеристиками, що вимагаються, у заданій області вхідних даних. Налагодження тісно стикається з тестуванням, передує йому і є по суті мистецтвом виявлення природи помилок й їхньої локалізації.

Методології й стратегія налагодження починається не по завершенні написання тексту програми, а безпосередньо в момент написання. Для легкості наступного налагодження рекомендується писати невеликі, добре закоментовані й специфіковані модулі, які виконують вузькі функції.

Під час налагодження треба постаратися визначити природу помилки, чи є це помилка помилкою апаратури, операційної системи (що малоймовірно), компілятора (читайте інструкції) або власної програми. Якщо помилка в програмі, постарайтеся визначити модуль, у якому вона виникає, виключите з розгляду найменш імовірні джерела помилки, спробуйте звузити область пошуку. Перевірте, чи є помилка повторюваної або стійкою.

Як правило найбільші труднощі з налагодженням зазнають програмісти, що пробують звалити провину на апаратуру, операційну систему, компілятор, і інші "зовнішні" причини. Будьте скурпульозні, методичні й логічні в пошуку помилок. Застосовуйте систематичний підхід до пошуку помилки. Насамперед, перевіряйте найпростіші припущення, у них легше й швидше знайти помилку або переконатися в її відсутності. Нічого не приймайте на віру. Наприклад, доводи "це дуже простий модуль, у ньому не може бути помилки", "зміни в працюючому модулі були мінімальні", "ще вчора це працювало правильно" і тому подібні не є гарантією відсутності помилки. Частіше використовуйте при роботі сіру речовину, що знаходиться у вашій голові. Розмовляйте зі своїми друзями. Дуже часто помилка лежить на поверхні, і ви просто "придивилися" до програми. Свіжий погляд, іноді, дозволяє не тільки виявити помилку, але й підвищити ефективність роботи коду. Ще одним, добре забутим, способом налагодження є включення в програму елементів, які роздруковують значення підозрюваних

змінних, видають повідомлення про нормальну роботу того або іншого модуля. Проте, найкращими критеріями налагодження є досвід і терпіння.

6.6. Структури даних у Паскалі

Мова Паскаль належить до парадигми структурного програмування. Це значить, що поняття «структура» входить у програму не тільки на рівні загальної її побудови, але й передбачає структурування самої логіки й даних, які обробляються. Насамперед, даний принцип знайшов своє застосування в типізації мови, тобто наділенні її строгою системою визначених типів, комбінуючи які програміст міг створювати свої власні типи, причому вже не тільки прості (атомарні), але й складні конструкції, що представляють собою структури даних.

У Паскалі використовуються такі структурні типи :

- Масиви
- Записи
- Файли

Масив - це одно- або багатомірна таблиця даних одного типу. Кожен осередок таблиці має свій індекс (в одномірному випадку) або набір індексів (у багатомірному). Масив називають структурою даних з випадковим доступом, оскільки до будь-якого елемента масиву можна звернутися, просто вказавши його індекси, тобто всі елементи однаково доступні в будь-який момент часу. Масив визначається, насамперед, загальним типом його елементів й їхньою кількістю. Кількість елементів масиву, у свою чергу, визначається кількістю індексів і діапазоном їхньої зміни. При описі змінної типу масив під кожен елемент виділяється фіксований об'єм пам'яті , що і є головним недоліком цієї структури, по-перше, тому що в деяких системах програмування Паскаль під змінну виділяється обмежена кількість пам'яті, що не дозволяє використовувати масиви більших розмірів, по-друге, тому що така організація не дає можливості

змінювати розмір масиву, що приводить до певного роду незручностей, коли доводиться марно копіювати більші об'єми пам'яті, наприклад, при передачі параметра-масиву підпрограмі.

Запис - зв'язана структура, що складається з декількох елементів (полів) різних (можна й однакових) типів. По суті, запис дуже схожий на одномірний масив, але з елементами різних типів, крім того, доступ до конкретного поля запису здійснюється вже не через індекс, а вказівкою ідентифікатора (тобто імені) цього поля.

Файл - динамічна структура даних, розмір якої може мінятися в процесі виконання над ним яких-небудь дій (він може дорівнювати нулю, що відповідає порожньому файлу). Файл - структура, що складається з послідовності компонентів одного типу. Властивості послідовності визначає послідовний доступ до елементів, тобто в кожен момент часу може бути доступний тільки один елемент файлу. Основна операція над файлами - конкатенація (або злиття) файлів - дозволяє створювати файли необмеженої довжини.

Залежно від способу оголошення можна виділити три види файлів:

- типізовані файли (задаються реченням FILE OF...);
- текстові файли (визначаються типом TEXT);
- нетипізовані файли (визначаються типом FILE).

Наприклад
```
var  f1 : file of char;
     f2 : text;
     f3 : file;
```

Текстові файли трактуються як послідовності рядків змінної довжини. Наприкінці кожного рядка ставиться спеціальна ознака кінця рядка EOLN. Типізовані файли - послідовності компонент визначеного типу. Довжина будь-якого компонента типізованого файлу строго постійна, що дає можливість організувати прямий доступ до кожного з них (тобто доступ до компонента по його порядковому номеру). Перед першим звертанням до процедур

вводу-виводу вказівник файлу стоїть в його початку й вказує на перший компонент із номером 0. Після кожного читання або запису вказівник зрушується до наступного компонента файлу. Змінні в списках вводу-виводу повинні мати той же тип, що й компоненти файлу. Якщо цих змінних у списку трохи, вказівник буде зміщатися після кожної операції обміну даними між змінними й дисковим файлом.

Файли стають доступні програмі тільки після виконання особливої процедури відкриття файлу. Ця процедура полягає у зв'язуванні раніше оголошеної файлової змінної з ім'ям існуючого або знову створюваного файлу, а також у зазначенні напрямку обміну інформацією: читання з файлу або запис у нього. Файлова змінна зв'язується з ім'ям файлу в результаті звертання до стандартної процедури ASSIGN:

ASSIGN (<ф.з.>, <ім'я файлу>); .

Тут <ф.з. > - файлова змінна (правильний ідентифікатор, оголошений у програмі як змінна файлового типу);

<ім'я файлу> - текстове вираження, що містить ім'я файлу.

Ім'я файлу - це будь-яке вираження строкового типу, що будується за загальними правилами визначення імен :

ім'я містить до восьми дозволених символів;

ім'я починається з будь-якого дозволеного символу;

за ім'ям може іти розширення - послідовність до трьох дозволених символів; розширення, якщо воно є, відокремлюється від імені крапкою. Перед ім'ям може вказуватися так званий шлях до файлу: ім'я диска й/або ім'я поточного каталогу й імена каталогів вищестоящих рівнів. Наприклад:

```
var
finp: text;
fout:file of String;
const
```

```
name = 'c:\XT06\out.txt';
begin
        assign(finp,'123.dat') ;
        assign(fout,name);
end.
```

Ініціювати файл означає вказати для цього файлу напрямок передачі даних. У Паскалі можна відкрити файл для читання, для запису інформації, а також для читання й запису одночасно. Для читання файл ініціюється за допомогою стандартної процедури RESET:

RESET (<ф.п.>); Тут <ф.з.> - файлова змінна, зв'язана раніше процедурою ASSIGN із уже існуючим файлом. При виконанні цієї процедури спеціальна змінна- вказівник, пов'язана із цим файлом, буде вказувати на початок файлу, тобто на компонент із порядковим номером 0.

Стандартна процедура REWRITE (<ф.з.>) ініціює запис інформації у файл. Процедурою REWRITE не можна ініціювати запис інформації в раніше існуючий дисковий файл: при виконанні цієї процедури старий файл знищується й ніяких повідомлень про це в програму не передається. Новий файл підготовляється до прийому інформації і його вказівник приймає значення 0.

Стандартна процедура APPEND (<ф.з. >) ініціює запис у раніше існуючий текстовий файл для його розширення, при цьому вказівник файлу встановлюється в його кінець. Процедура APPEND застосовна тільки до текстових файлів, тобто їх файлова змінна повинна мати тип TEXT.

Процедура CLOSE (<ф.з.>) Закриває файл, однак зв'язок файлової змінної з ім'ям файлу, установлений раніше процедурою ASSIGN, зберігається.

Процедура RENAME (<ф.з.>, <нове ім'я>) перейменовує файл Тут <нове ім'я> - строковий вираз, що містить нове ім'я файлу. Перед виконанням процедури необхідно закрити файл.

Процедура ERASE (<ф.з.>) знищує файл. Перед виконанням процедури

необхідно закрити файл.

Функція EOF (<ф.з. >) : BOOLEAN. Логічна функція, яка тестує кінець файлу. Повертає TRUE, якщо файловий вказівник стоїть наприкінці файлу. При записі це означає, що черговий компонент буде доданий у кінець файлу, при читанні -що файл вичерпаний.

Процедура READ. Забезпечує читання чергових компонентів типізованого файлу. Формат звертання:

READ (<ф.з.>,<сп.вводу>)

Тут <сп.вводу> - список вводу, що містить одну або більше змінних такого ж типу, що й компоненти файлу. Файлова змінна <ф.з.> повинна бути оголошена реченням FILE OF... і пов'язана з ім'ям файлу процедурою ASSIGN. Файл необхідно відкрити процедурою RESET. Якщо файл вичерпаний, звертання до READ спричинить помилку вводу-виводу.

Процедура WRITE. Використається для запису даних у типізований файл. Формат звернення:

WRITE (<ф.з.>,<сп.виводу>)

Тут <сп.виводу> - список виводу, що містить одне або більше виразів того ж типу, що й компоненти файлу.

Процедура SEEK. Зміщає вказівник файлу до необхідного компонента. Формат звернення:

SEEK (<ф.з.>,<N компонента>)

Тут <N компонента> - вираз типу LONGINT, що вказує номер компонента файлу. Перший компонент файлу має номер 0. Процедуру не можна застосовувати до текстових файлів.

Функція FILESIZE. Повертає значення типу LONGINT, що містить кількість компонентів файлу. Формат звернення:

FILESIZE (<ф.з.>)

Функцію не можна використовувати для текстових файлів. Щоб

перемістити вказівник у кінець типізованого файлу, можна написати:

seek (FileVar, FileSize(FileVar));

де FILEVAR - файлова змінна.

Функція.FILEPOS. Повертає значення типу LONGINT, що містить порядковий номер компонента файлу, що буде оброблятися наступною операцією вводу-виводу. Формат звернення:

FILEPOS (<ф.з.>)

Функцію не можна використати для текстових файлів. Перший компонент файлу має порядковий номер 0.

Приклад роботи зі структурованими типами:

Вихідні дані задані у вигляді наступної таблиці

Назва деталі	Ф. И. О. робітника	Випущено	ціна 1 деталі

Обчислити:

1. Суму, на яку випущені деталі кожним робітником

```
procedure punkt1;
var
  a:tabl;
  b:tabl1;
begin
      reset(f1);        rewrite(f2);
      while not eof(f1) do
                        begin
                        read(f1,a);
                        b.vip:=a.kol*a.st;
                        b.fio:=a.fio;
                        write(f2,b);
                       end;
      close(f1); close(f2);
end;
```

2. Прибуток від випуску всіх деталей

```
procedure punkt2;
var
```

```
  s:real;
  b:tabl1;
begin
      reset(f2);
      s:=0;
      while not eof(f2) do
                                 begin
                                  read(f2,b);
                                s:=s+b.vip;
                                end;
      close(f2);
      writeln('stoimost vsex detaley=',s:7:2);
end;
```

Підпрограма створення вихідного файлу

```
procedure formfile;
 var
  a:tabl;
  key:char;
 begin
      rewrite(f1);
      key:='y';
      while key='y' do
          begin
          write('vvesti naim   '); readln(a.naim);
          write('vvesti fio   '); readln(a.fio);
          write('vvesti kol   '); readln(a.kol);
          write('vvesti st   '); readln(a.st);
          write(f1,a);
          write('vvesti zapis [y/n]'); readln(key);
          end;
      close(f1);
 end;
```

Підпрограма перегляду результуючого файлу

```
procedure prosmotr;
var
  b:tabl1;
begin       reset(f2);
       writeln('!-i-i-i-i-i!-i-i-i-i-i!');
      writeln('!   fio    !  summa   !');
       writeln('!-i-i-i-i-i!-i-i-i-i-i!');
      while not eof(f2) do
                        begin
```

```
                        read(f2,b);
                        writeln('!',b.fio:10,'!',b.vip:10:3,'!');
                       end;
       writeln('!-i-i-i-i-i!-i-i-i-i-i!');
       close(f2);
end;
```

Основна програма

```
program menu;
uses CRT;
type
    tabl=record
            naim:string[10];
            fio:string[10];
            kol:integer;
            st:real;
         end;
   tabl1=record
            fio:string[10];
            vip:real;
         end;
    ftabl=file of tabl;
    ftabl1=file of tabl1;
var
  a:tabl1;
  f1:ftabl;
  f2:ftabl1;
  r:integer;
{тексти процедур}
   begin
  clrscr;
  assign(f1,'E:\file.isx');    assign(f2,'E:\file.rez');
  while true do
          begin
          writeln('******************');
          writeln('1: sozdanie fajla');
          writeln('2: summa');
          writeln('3: dohod');
          writeln('4: prosmotr');
          writeln('5: exit');
          write('rezim raboty ?'); readln(r);
          case r of
               1:formfile;
               2:punkt1;
```

```
            3:punkt2;
            4:prosmotr;
            5:halt;
            end;
        end;
end.
```

Список використаної літератури

1. С.Гудман, С.Хидетниєми. Введення в розробку й аналіз алгоритмів - Москва ,Мір ,1981р.-364 с.
2. А.Ахо, Дж.Хопкрофт, Дж.Ульман. Побудова й аналіз обчислювальних алгоритмів- Москва , Мир , 1979 р.-535с.
3. В. Турский. Методологія програмування-москва ,Мір ,1981р.-263 с.
4. Інформатика: Базовий курс / С.В.Симонович й ін. - Пітер, 2002.- 640с.
5. Методичний посібник для виконання курсової роботи з дисципліни «Основи інформаційних технологій і програмування»/Сост.: В.Г.Суслова, И.В.Диннік - Донецьк, ДонНТУ, 2005-68с.
6. Архангельський А.Я. Мова Pascal й основи програмування в Delphi. Навчальний посібник - М.:ТОВ “Біном^-Пресс” ,2004.- 496 с.
7. Методична допомога з побудови алгоритмів і проектуванню програм / Сост.: И.Н. Кузык-Артамонова, И.В. Диннік - Донецьк: ДонНТУ, 2006 - 57 с.
8. Методичний посібник з обчислювальної математики і програмування / Укладачі: І.В. Диннік, О.А.Тіхонова - Донецьк: ДонНТУ, 2008 - 71 с.

Додаток 1. Приклад оформлення титульної сторінки

ДЕРЖАВНИЙ ВИЩІЙ НАВЧАЛЬНИЙ ЗАКЛАД
МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ
ДОНЕЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Кафедра "Обчислювальної
математики і програмування"

ПОЯСНЮВАЛЬНА ЗАПИСКА

до курсовій роботі з_____

Тема роботи :"xxxxxxxxxxxxxxxxxx"

Студента групи XXX-09
Іванова І.І.

Керівник роботи
Петров П.П.

Донецьк-2009

Додаток 2. Приклад оформлення листа завдання

ДЕРЖАВНИЙ ВИЩІЙ НАВЧАЛЬНИЙ ЗАКЛАД
МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ
ДОНЕЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Факультет_____
Кафедра_____
Спеціальність_____

Завдання

на курсову роботу студентові

Іванову Івану Івановичу

1.Тема роботи :”xxxxxxxxxxxxxxxxxx”

2. Термін здачі студентом закінченої роботи ___ _____ 2008 р.

3.Вихідні дані до роботи :

Керівник роботи _____ _____
Підпис Ф И О

Донецьк-2009

Додаток 3. Приклад оформлення календарного плану

КАЛЕНДАРНИЙ ПЛАН

Номер етапу	Найменування етапу виконання курсової роботи	Термін виконання	Примітка

Студент Іванов І.І.

Керівник Петров П.П.

Додаток 4. Приклад оформлення реферату

РЕФЕРАТ

ТАБЛИЦЯ, АЛГОРИТМ, БЛОК-СХЕМА, ГРАФІКИ ЗАЛЕЖНОСТЕЙ

сторінок -30 ,таблиця -1, джерел -3 , додатків -2.

Об'єктом дослідження є ... …… …… …… …… ……..... У роботі проводиться розрахунок ... …… …… …… …… ……...

Додаток 5. Приклад оформлення змісту

ЗМІСТ

Додаток 6. Приклад оформлення списку використаної літератури

Список використаної літератури

1. Черняк И.Л., Яруник С.А., Бурчаков Ю.И. Технологія і механізація підземного видобутку вугілля: підручник для вузів. – М.:Надра,1981.

2. Виробничі процеси в очисних вибоях поліпшених шахт: Навчальний посібник для вузів/И.Ф. Ярембаш, В.Д. Мороз, И.С. Костюк. Під загальною редакцією И.Ф. Ярембаша. – Донецьк: Донгту,1999

3. Сергєєв Н.П., Доминин Л.Н. Алгоритмізація і програмування: Навчальний посібник для вузів: - М.:Радіо і зв'язок, 1982. – 232 с.

Додаток 7. Кодування псевдографіки у коді ASCII

179	│	189	╜	199	╟	209	╤
180	┤	190	╛	200	╚	210	╥
181	╡	191	┐	201	╔	211	╙
182	╢	192	└	202	╩	212	╘
183	╖	193	┴	203	╦	213	╒
184	╕	194	┬	204	╠	214	╓
185	╣	195	├	205	═	215	╫
186	║	196	─	206	╬	216	╪
187	╗	197	┼	207	╧	217	┘
188	╝	198	╞	208	╨	218	┌

МЕТОДИЧНИЙ ПОСІБНИК ДЛЯ ВИКОНАННЯ КУРСОВОЇ РОБОТИ З ОБЧИСЛЮВАЛЬНОЇ МАТЕМАТИКИ ТА ПРОГРАМУВАННЯ

(учбово-методичний посібник для студентів спеціальностей ХТ і ТХВ)

Укладачі: І. В. Диннік, ст. викладач кафедри ОМіП

О.А. Тихонова, асистент кафедри ОМіП