Тема 7 ТЕХНОЛОГІЯ СТРУКТУРНОГО ПРОГРАМУВАННЯ

7.1. ПОНЯТТЯ СТРУКТУРИ ПРОГРАМИ

Структура програми — штучно виділені програмістом частини програми, що взаємодіють. Використання раціональної структури усуває проблему складності розробки; робить програму зрозумілою людям; підвищує надійність роботи програми при скороченні терміну її тестування та термінів розробки загалом.
Часто деяку послідовність інструкцій потрібно повторити у кількох місцях програми. Щоб програмісту не доводилося витрачати час і зусилля копіювання цих інструкцій, у більшості мов програмування передбачаються кошти на організацію підпрограм. Таким чином, програміст отримує можливість присвоїти послідовності інструкцій довільне ім'я та використовувати це ім'я як скорочений запис у тих місцях, де зустрічається відповідна послідовність інструкцій. Підпрограма — деяка послідовність інструкцій, яка може бути викликана в кількох місцях програми.
Опис підпрограми (функції чи процедури) складається з двох частин: заголовка та тіла. Заголовок містить ідентифікатор підпрограми. Тіло складається з однієї чи кількох інструкцій. Ідентифікатор підпрограми використовується як скорочений запис у тих місцях програми, де зустрічається відповідна послідовність інструкцій.
Навряд чи варто було докладно говорити про таку просту форму запису, якби за нею не ховалися важливі та основні поняття. Насправді процедури та функції, які називаються підпрограмами, є одним з тих небагатьох фундаментальних інструментів у мистецтві програмування, які надають вирішальний вплив на стиль та якість роботи програміста.
Процедура — це спосіб скорочення програмного тексту, а й, що важливіше, засіб розкладання програми на логічно пов'язані, замкнуті елементи, що визначають її структуру. Розкладання на частини суттєво для розуміння програми, особливо якщо програма складна і важко доступна для огляду через велику довжину тексту. Розкладання на підпрограми необхідне як документування, так верифікації програми. Тому бажано оформляти послідовність інструкцій як підпрограми, навіть якщо підпрограма використовується одноразово і, отже, відсутня мотив, що з скороченням тексту програми.
Додаткова інформація про змінні (які передаються та використовуються у процедурі) або про умови, яким мають задовольняти аргументи, задається у заголовку процедури. Про корисність процедури, зокрема про її роль під час структуризації програми, незаперечно свідчать ще два поняття у програмуванні. Деякі змінні (їх зазвичай називають допоміжними або локальними змінними), які використовуються всередині процедури, не мають сенсу за її межами. У програмі значно простіше розібратися, якщо очевидно вказані області дії таких змінних. Процедура постає як природна текстова одиниця, з допомогою якої обмежується сфера існування про локальних змінних.
Ймовірно, найбільш загальна тактика програмування полягає у розкладанні процесу на окремі дії: функціонального опису на підфункції, а відповідних програм – на окремі інструкції. На кожному такому кроці декомпозиції слід переконатися, що розв'язання приватних завдань призводять до вирішення спільного завдання; обрана послідовність окремих дій розумна; обрана декомпозиція дозволяє отримати інструкції, в якомусь сенсі ближчі до мови, якою буде реалізована програма.
Остання вимога виключає можливість прямолінійного просування від початкової постановки завдання до кінцевої програми, яка має вийти зрештою. Кожен етап декомпозиції супроводжується формулюванням приватних підпрограм. У процесі роботи може виявитися, що обрана декомпозиція

невдала у тому сенсі хоча б оскільки підпрограми незручно висловлювати з допомогою наявних коштів. У цьому випадку один або кілька попередніх кроків декомпозиції слід переглянути заново.
Якщо бачити в поетапній декомпозиції та одночасному розвитку та деталізації програми поступове просування вглиб, то такий метод при вирішенні завдань можна охарактеризувати як низхідний (згори донизу). І навпаки, можливий такий підхід до вирішення задачі, коли програміст спочатку вивчає обчислювальну машину та/або мову програмування, що є в його розпорядженні, а потім збирає деякі послідовності інструкцій в елементарні процедури, типові для розв'язуваного завдання. Елементарні процедури використовуються на наступному рівні ієрархії процедур. Такий метод переходу від примітивних машинних команд до необхідної реалізації програми називається висхідним (знизу догори).
На практиці розробку програми ніколи не вдається провести строго в одному напрямку (згори донизу або знизу догори). Проте за конструюванні нових алгоритмів низхідний метод зазвичай домінує. З іншого боку, при адаптації програми до дещо змінених вимог перевага найчастіше надається висхідному методу.
Обидва методи дозволяють розробляти програми, яким властива структура — властивість, що відрізняє їхню відмінність від аморфних лінійних послідовностей інструкцій чи команд машини. І надзвичайно важливо, щоб використовувана мова повною мірою відображала цю структуру. Тільки тоді остаточний вид одержаної програми дозволить застосувати систематичні методи верифікації.

7.2. МОДУЛЬ ТА ОСНОВНІ ПРИНЦИПИ СТРУКТУРНОГО ПІДХОДУ

7.2.1. Поняття модуля

Якщо програма розбивається на підпрограми, то представлення результатів і аргументів часто доводиться вводити нові змінні і таким чином встановлювати зв'язок між підпрограмами. Такі змінні слід вводити та описувати на тому етапі розробки, на якому вони були потрібні. Більш того, деталізація опису процесу може супроводжуватися деталізацією опису структури змінних, що використовуються. Отже, в мові мають бути засоби для відображення ієрархічної структури даних. Зі сказаного видно, яку важливу роль грає при покроковій розробці програми поняття процедури, локальності процедур і даних, структурування даних.
Проектування починається із фіксації зовнішніх специфікацій. З зовнішніх специфікацій складається опис внутрішнього алгоритму програми, обов'язково з структурою внутрішніх даних. Далі великі функції розбиваються на підфункції до досягнення підфункції розміру модуля - підпрограми (процедури чи функції) мови програмування, яких пред'являються особливі додаткові вимоги.
Модуль- фундаментальне поняття та функціональний елемент технології структурного програмування.
Модуль -це підпрограма, але оформлена відповідно до особливих правил.
*Правило 1.*Модуль повинен мати один вхід і один вихід і виконувати строго однозначну функцію, яка описується простою поширеною пропозицією природної (російської) мови або навіть пропозицією без присудка.
*Правило 2*Модуль повинен забезпечувати компіляцію, незалежну від інших модулів, із «забуттям» усіх внутрішніх позначень модулів.
*Правило 3*Модуль може викликати інші модулі за їхніми іменами.
*Правило 4*Хороший модуль не використовує глобальні змінні спілкування з іншим модулем, оскільки потім важко знайти модуль, який псує дані. Якщо все ж таки використовуються глобальні змінні, то потрібно чітко коментувати ті модулі, які тільки читають, і ті модулі, які можуть змінювати дані.
*Правило 5*Модуль кодується лише стандартними структурами та ретельно коментується.
У поняття структури програми включаються склад та опис зв'язків всіх модулів, які реалізують самостійні функції програми та опис носіїв даних, що беруть участь в обміні як між окремими підпрограмами, так і введені та виведені з/на зовнішніх пристроїв.
У разі складної великої програми необхідно опанувати спеціальні прийоми отримання раціональної структури програми. Раціональна структура програми забезпечує майже дворазове скорочення обсягу програмування та багаторазове скорочення обсягів та термінів тестування, а отже, принципово знижує витрати на розробку.

Підпорядкованість модулів зручно зображати схемою ієрархії. Схема ієрархії відбиває лише підпорядкованість підпрограм, але з порядок їх виклику чи функціонування програми. Схема ієрархії може мати вигляд, показаний на рис. 7.1.

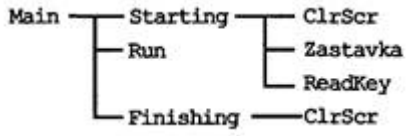


*Рис. 7.1.*Схема ієрархії простої програми

Така схема ієрархії зазвичай доповнюється розшифровкою функцій, що виконуються модулями.
До складання схеми ієрархії доцільно скласти зовнішні специфікації програми та скласти функціональні описи програми разом з описом змінних носіїв даних. Особливу увагу треба приділити ієрархії типів структурованих даних та їхнього коментування. Декомпозиція програми на підпрограми провадиться за принципом від загального до приватного, більш детального. Взагалі процес складання функціонального опису та складання схеми ієрархії є ітераційним. Вибір найкращого варіанта є багатокритеріальним. Розчленування повинне забезпечувати зручний порядок введення частин в експлуатацію.
Реалізація програми (кодування, складання, тестування) має вестися за розробленим заздалегідь планом і починатися з верхніх модулів схеми ієрархії. Відсутні модулі нижніх рівнів замінюються заглушками, які є найпростішими підпрограмами: або без дій; або вхідні дані, що виводять у файл налагодження; або тестові дані, що повертають у вищестоящі модулі (які зазвичай присвоюються всередині заглушки); або містять комбінацію цих процесів.
Структурний підхід до програмування є як методологією, і технологію створення програм. Свого часу його використання забезпечило підвищення продуктивності праці програмістів під час написання та налагодження програм; отримання програм, які складаються з модулів та придатні для супроводу; створення програм колективом розробників; закінчення створення програми у заданий термін.
Структурний підхід до програмування сприйняв і використовує багато методів в галузі проектування складних технічних систем. Серед них блочно-ієрархічний підхід до проектування складних систем, стадійність створення програм, низхідне проектування, методи оцінки та планування.
Структурний підхід рекомендує дотримуватись наступних принципів при створенні програмного виробу:
- Модульність програм;
- Структурне кодування модулів програм;
- низхідне проектування раціональної ієрархії модулів програм;
- низхідна реалізація програми з використанням заглушок;
- Здійснення планування на всіх стадіях проекту;
- Наскрізний структурний контроль програмних комплексів в цілому і складових їх модулів.

Модульність програм характеризується тим, що програма складається з модулів. Деякі смислові групи модулів зосереджуються на окремих файлах. Наприклад, в окремих файлах (Unit) можуть бути зосереджені модулі текстового редактора та модулі ієрархічного меню.
Структурне кодування модулів програм полягає у особливому оформленні їх текстів. У модуля має бути легко помітний заголовок з коментарем, що пояснює функціональне призначення модуля. Імена змінних мають бути мнемонічними. Суть змінних та порядок розміщення в них інформації мають бути пояснені коментарями, а код модуля закодований з використанням типових алгоритмічних структур із використанням відступів.
Низхідне проектування раціональної ієрархії модулів програм полягає у виділенні спочатку модулів найвищого рівня ієрархії, а потім підлеглих модулів.
Низхідна реалізація програми полягає у первинній реалізації групи модулів верхніх рівнів, які називаються ядром програми, і далі поступово, відповідно до плану, реалізуються модулі нижніх рівнів. Необхідні для лінкування програми, відсутні модулі імітуються заглушками.
Здійснення планування усім стадіях проекту дозволяє спочатку спланувати як склад стадій, і тривалість всіх етапів робіт. Таке планування дозволяє завершити розробку в заданий термін за заданих витрат на

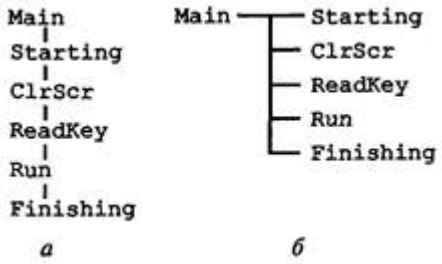
розробку. Далі плануються порядок і час інтеграції модулів у ядро, що все розширюється. Плануються заходи щодо тестування програми від ранніх до заключних етапів.
Наскрізний структурний контроль полягає у дотриманні наперед наміченого плану тестування, який охоплює період від розробки зовнішніх специфікацій, далі внутрішніх специфікацій та їх коригування в періоді реалізації аж до приймально-здавальних випробувань. Модулі, що складають програму, тестуються як під час написання їх коду, так і при автономному тестуванні, інспекції їх вихідного коду, при тестуванні відразу після підключення до ядра.

7.2.2. Поняття заглушки модуля

При структурному програмуванні програма переважно реалізується (збирається і тестується) зверху донизу. Спочатку із 20—30 модулів пишеться ядро. Щоб почати тестувати, модулі нижніх рівнів замінюються заглушками. Після закінчення тестування ядра, заглушки замінюються новими готовими модулями, але якщо програма ще не закінчена, то для успішного її лінкування знадобляться все нові заглушки модулів, що відсутні. Тепер можна приступати до тестування зібраної частини тощо.
Заглушка- Це макет модуля. Найпростіша заглушка – це підпрограма чи функція без дій. Більш складна заглушка може виводити повідомлення про те, що відпрацював такий модуль. Ще більш складні заглушки можуть виводити вхідну інформацію в файл налагодження. Нарешті, ще складніші заглушки видають на вихід тестову інформацію, необхідну перевірки вже реалізованих модулів.
Написання заглушок — «зайва» робота, але потрібне мистецтво проектувальника, щоб максимальна кількість заглушок була простою, а тестування вже зібраної частини програми було б повним.

7.2.3. Засоби зміни топології ієрархії програми

У процесі низхідного проектування раціональної ієрархії модулів програми необхідно отримати оптимальну підпорядкованість.
Схемі ієрархії можна надати будь-який топологічний малюнок. Так, схемою ієрархії, зображеної на рис. 7.2, а можна надати вигляд, зображений на рис. 7.2, б.
Фрагменти з вертикальними викликами можуть бути перетворені на виклики з одного рівня за допомогою введення додаткового модуля, який може не виконувати жодних корисних функцій з точки зору алгоритму програми. Функція нового модуля може перебувати лише моніторингу, т. е. виклику інших модулів у порядку.



Мал. 7.2. Схема ієрархії однієї й тієї програми: а — з вертикальними викликами; б - з горизонтальними викликами

Фрагменти з горизонтальними викликами на одному рівні можуть бути перетворені на вертикальні виклики модулів різних рівнів за допомогою введення додаткових змінних. Ці змінні не можуть бути отримані декомпозицією функціонального опису на підфункції. Ці додаткові змінні мають тип цілий чи логічний і називаються прапорами (семафорами, ключами) подій. Їх зміст зазвичай характеризується фразою: «Залежно від передісторії дій, виконати такі дії».
У процесі проектування необхідно створити кілька проектних ітерацій, генеруючи щоразу нову схему ієрархії, і порівняти ці ієрархії за критеріями оцінки якості схеми ієрархії.

7.2.4. Критерії оцінки якості схеми ієрархії

Проектування структури програми передує розробка зовнішніх функціональних описів. р align="justify"> Функціональні описи (алгоритми виконання програми) для досягнення їх сприйняття повинні бути декомпозіровані від загального до приватного. Також вони повинні включати описи форм подання та обсягу внутрішніх даних.
Отже, спочатку є перший варіант схеми ієрархії, отриманий шляхом простого членування функцій програми на підфункції із зазначенням змінних, необхідних розміщення даних різних кроках обробки. Скоріше цей варіант не є оптимальним і потрібні проектні ітерації (зазвичай виконуються методом «проб і помилок») для покращення топології схеми.
Кожен новий варіант порівнюється з попереднім варіантом за описаними тут критеріями. Генерація варіантів припиняється за неможливості подальших поліпшень.
Фонд критеріїв оптимальності схем ієрархії є необхідною підмогою при оптимізації схем ієрархії та складається з 13 критеріїв.
Перший- Повнота виконання специфікованих функцій.
Другий -можливість швидкого та дешевого поповнення новими, раніше не специфікованими функціями.
Третій, що випливає з блочно-ієрархічного підходу, — доступність для проектувальника складових частин програми.
Четвертий критерій оцінки якості структури - максимальна незалежність за даними окремих частин програми.
П'ятий можливість пов'язування програми з великою багаторівневою схемою ієрархії конкретним редактором зв'язків (лінковщиком). Якщо починаєте працювати над новою програмою, то дуже корисно виконати на ЕОМ її модель у вигляді порожніх заглушок модулів, які не містять жодних дій.
Шостий- Достатність оперативної пам'яті. Тут розглядаються варіанти з описом особливо структурованих статичних і динамічних змінних різних рівнях схеми ієрархії. Перевірка задоволення цього критерію здійснюється розрахунками з деякими машинними експериментами.
Сьомий- Оцінка впливу топології схеми ієрархії на швидкість виконання програми при використанні оверлеїв (динамічного завантаження програми) і механізму підкачування сторінок при розробці програми, яка цілком не може бути розміщена в оперативній пам'яті.
Восьмий -відсутність різних модулів, що виконують подібні дії. В ідеалі — той самий модуль викликається різних рівнях схеми ієрархії.
Дев'ятий - Досягнення при реалізації програми такого мережевого графіка роботи колективу програмістів, який забезпечує рівномірне завантаження колективу за ключовими датами проекту.
Десятий усіляке скорочення витрат на тестування вже зібраного ядра програми за ключовими датами мережевого графіка реалізації. Характеризується простотою заглушок і якістю тестування по всіх обчислювальних маршрутах модулів. Досягається первинної реалізацією зверху донизу модулів введення та виведення програми з відстрочкою реалізації інших гілок схеми ієрархії. Зазвичай витрати на тестування як за термінами, так і грошима становлять близько 60% вартості всього проекту. Хороша схема ієрархії скорочує витрати на тестування проти початковим варіантом в 2—5 разів і більше.
Одинадцятий- використання в даному проекті якомога більшої кількості розроблених у попередніх проектах модулів і бібліотек при мінімальному обсязі частин, що виготовляються заново.
Дванадцятий— вдалий розподіл модулів за файлами програми та бібліотекам, що компілюються.
Тринадцятий -накопичення вже готових модулів та бібліотек модулів для використання їх у нових розробках.
Отже, хороша структура програми забезпечує скорочення загального обсягу текстів у 2 чи 3 рази, що відповідно здешевлює проект; на кілька порядків здешевлює тестування (на тестування зазвичай припадає щонайменше 60 % від загальних витрат проекту), і навіть полегшує і здешевлює супровід програми.

7.2.5. Рекомендації щодо організації процесу розробки схеми ієрархії

Як правило, складання зовнішніх, потім внутрішніх функціональних описів і надалі структури програми здійснює група від двох до семи кваліфікованих програмістів - системних аналітиків.
Окремі варіанти структури програми розробляються до можливості їх порівняння. При цьому використовуються такі документи:
• опис алгоритму (функціонування) програми чи методів розв'язання задачі разом із описом даних;

• схема ієрархії модулів програми з розшифровкою позначень та функцій модулів;
• паспорти модулів.
На підставі цих документів готуються опис алгоритму програми з урахуванням модульного поділу та мережевий графік реалізації та тестування програми з тестами, складеними до програмування.
Паспорт модуля— внутрішній документ проекту, що є конвертом з ім'ям модуля. Усередині конверта містяться описи: прототипи виклику самого модуля та модулів, що викликаються модулем даних; розшифровки вхідних та вихідних змінних модуля; опис функції, що виконується модулем; принципів реалізації алгоритму модуля з описом основних структур даних У паспорті модуля можуть бути копії літературних джерел з описами основних ідей алгоритму. У процесі виконання проекту паспорт модуля коригується до технічного завдання розробці цього модуля, а після реалізації — до опису модуля.
Група системних аналітиків перевіряє загальну однозначність цих описів і генерує нові варіанти схем ієрархії. При реалізації ця група тестує вже зібране ядро програми з усіх нових ключових дат мережного графіка проекту. У процесі тестування ядра програми з допомогою заглушок уточнюються діапазони даних. У разі потреби системні аналітики коригують паспорти модулів перед програмуванням модулів за результатами уточнення діапазонів даних.
Найголовніше у схемі ієрархії — мінімізація зусиль зі збирання та тестування програми. При використанні заглушок можна добре тестувати сполучення модулів, але не самі модулі. Тестування самих модулів вимагатиме витончених складних заглушок та астрономічної кількості тестів. Вихід - до інтеграції модулів тестувати модулі з використанням провідних програм. Також рекомендується здійснювати реалізацію з деяким порушенням принципу «зверху-вниз». Рекомендується спочатку з дотриманням принципу «зверху-вниз» реалізувати галузь схеми ієрархії, що відповідає за введення інформації з перевіркою її коректності, заглушивши гілки розрахунків та виведення на найвищому рівні. Далі реалізується галузь виведення інформації й у останню чергу — галузь розрахунків (функціонування програми). Якщо функцій програми багато, можна спочатку реалізувати модулі вибору функцій, заглушивши модулі самих функцій, і далі реалізовувати гілка кожної функції послідовно з дотриманням принципу «зверху—вниз».
Схема ієрархії повинна включати максимальну кількість модулів інших розробок. Багато модулів можна використовувати в інших розробках, однак це не стосується обчислювальних модулів, для яких через похибку рахунку можуть не підійти діапазони даних.
Тут дуже важливим є складання зручного графіка роботи з урахуванням планування загальної кількості програм кодувальників та їх рівномірного завантаження за термінами проекту, а також закінчення проекту у призначений термін.
Схема ієрархії повинна відбиватися на файли з вихідними текстами програм таким чином, щоб кожен файл містив якомога більше готових функцій із загальним призначенням. Це бажано для полегшення їх використання у подальших розробках.
Таким чином, окрім отримання грошей від замовника за розробку, програміст зобов'язаний підвищувати свій інтелектуальний капітал також за гроші замовника.
Існує дуже багато автоматизованих систем формування декомпозиції схем ієрархії, наприклад HIPO, SADT, R-TRAN.

7.3. РЕТРОСПЕКТИВНЕ ПРОЕКТУВАННЯ ДЕМОНСТРАЦІЙНОЇ ПРОГРАМИ MCALC ФІРМИ «BORLAND INC.»

Згідно з ретроспективно проведеним системним аналізом (див. тему 2), фірма «Borland Inc.» ухвалила рішення про реалізацію демонстраційного прикладу програми електронної таблиці. Цілком можливо згенерувати безліч варіантів реалізації електронної таблиці, починаючи від варіанта з усіма клітинами в одному вікні та закінчуючи, наприклад, варіантом Excel. Проте фірма Borland Inc. обрала варіант із прокручуванням інформації клітин у вікні, зміною адрес клітин при вставках рядків і стовпців, а також при їх видаленні. У проект запроваджено вимоги розробки некомерційного виробу. Розмір таблиці обмежений 100-100 клітинами. У програмі відсутня функція копіювання клітин. Вибрана складність варіанта, що реалізується, відповідає багато файловому проекту. Програма має функції підтримки виведення на екран, введення з клавіатури; у ній реалізований інтерпретатор формул із математичними функціями; для збереження інформації таблиці використовується файл складної організації, розглянутий гол. 3. Все це дає змогу продемонструвати можливості компілятора.

Програма Mcalc 1985-1988 гг. (Turbo Pascal 5.0) складається з наступних файлів:
• mcalc.pas - файл основної програми;
• mcvars.pas - файл глобальних описів;
• mcdisply.pas - файл підпрограм роботи з дисплеєм;
• mcmvsmem.asm - асемблерний файл підпрограм запам'ятовування в оперативній пам'яті інформації екрана, а також відновлення раніше збереженої інформації екрана;
• mcinput.pas - файл підпрограм введення даних з клавіатури;
• mcommand.pas — файл підпрограм, які обслуговують систему меню та дій, вибраних за допомогою меню;
• mcutil.pas - файл допоміжних підпрограм;
• mcparser.pas – файл інтерпретатора арифметичних виразів формул клітин.
Всі файли закодовані з дотриманням стандартів оформлення, що розвиваються. Так, у файлах mcdisply.pas, mcinput.pas описи прототипів підпрограм виконані з використанням більш раннього синтаксису мови програмування, що говорить про їхнє запозичення із програм, написаних раніше; при цьому можна виявити їхнє невелике модифікування (рефакторинг).
Хоча фірма Borland Inc. займається розробкою компіляторів, файл mcparser.pas також є запозиченим з UNIX YACC utility та лише частково модифікованим. Інші файли є оригінальними.
Асемблер файл mcmvsmem.asm є штучно доданим. Мета його додавання – демонстрація можливості використання асемблерних вставок. Алгорити, що містяться в ньому, цілком можна було б реалізувати мовою Pascal. Більше того, можна було б взагалі обійтися без реалізованих у ньому підпрограм, щоправда, було б видно деякі затримки виведення інформації на екран.
З метою здійснення нової проектної ітерації, що покращує проект, отримаємо з існуючого проекту проектну документацію, що складається з опису структури даних програми; функціонального опису основного ядра програми; схеми ієрархії модулів основного ядра програми; специфікації призначення модулів основного ядра програми.
Розглянемо організацію файлу mcvars.pas, що містить в основному опис структури внутрішніх даних програми. Файл містить описи в розділі interface. Секція впровадження порожня.
На початку файлу міститься код, який в залежності від наявності співпроцесора транслюється в одному з двох варіантів:

```
{$IFOPT N+}
{Є вбудований співпроцесор}
type
Real = Extended; {Заміна типу Real на Extended}
const
EXPLIMIT = 11356; {Гранове значення аргументу експоненти}
SQRLIMIT = 1E2466; {Гранічне значення аргументу SQRT}
....
{$ELSE}
const
{Тип Real не перевизначено}
EXPLIMIT = 88; {Гранове значення аргументу експоненти}
SQRLIMIT = 1E18; {Граничне значення аргументу SQRT}
....
{$ENDIF}
```

Описи констант містять такі блоки:
- Блок малих констант, що містять інформацію всіх виведених на екран і файли текстових написів (для русифікації всієї програми потрібно змінити тільки цю інформацію);
- блок парних рядків текстів меню та «гарячих» клавіш вибору тем меню;
- блок опису найважливіших констант, що визначають розмірність таблиці та розташування інформації на екрані

```
MAXCOLS = 100; { Maximum is 702 } {Розмір таблиці}
MAXROWS = 100;
MINCOLWIDTH = 3; {Мінімальна ширина стовпця}
MAXCOLWIDTH = 77; {Максимальна ширина стовпця}
....
```

- блок опису кольорів усіх полів екрану, модифікація констант якого дозволяє оперативно змінювати кольори;

— основні константи, мнемоніка імен яких полегшує сприйняття текстів програми

```
HIGHLIGHT = True; {Підсвічена поточна клітина}
NOHIGHLIGHT = False; {Непідсвічена клітина}
{Атрибут вмісту клітини}
TXT = 0;
VALUE = 1;
FORMULA = 2;
....
{Дозволені літери}
LETTERS: set of Char = ['A'..'Z', 'a'..'z'];
```

— коди клавіш клавіатури.

Слід зазначити, що наведені навіть коди клавіатури, що не використовуються в програмі керуючих клавіш. Це відповідає факту копіювання даних кодів із коду якоїсь іншої розробки.

Далі йдуть описи типу інформації вмісту табличної клітини та типу покажчика на клітину:

```
type
CellRec = record
Error: Boolean;
case Attrib : Byte of
TXT: (T: IString);
VALUE: (Value: Real);
FORMULA: (Fvalue: Real;
Formula: IString);
end;
CellPtr = ^CellRec; {Покажчик на клітку}
```

Даний тип організований так, що клітина завжди може містити ознаку помилки розрахунків Error та розміщувати три варіанти інформації: текст, значення та формулу.

Далі описано основні глобальні змінні. Описи починаються з визначення двовимірного масиву Cell покажчика, що постійно знаходиться в пам'яті, на клітини таблиці. Це дозволяє не витрачати пам'ять на порожні клітини. Пам'ять під інформацію клітини виділяється динамічно в кількості, що суворо відповідає інформації клітини. Без використання пам'яті, що динамічно виділяється, було б неможливо розмістити інформацію клітин таблиці в 640К пам'яті машин того часу.

```
MAXCOLS*MAXROWS*[SizeOf(Istring)+SizeOf(Extened)] = 100*100*[80+10] = 900K
```

Далі йдуть опис змінної, яка є покажчиком на поточну клітину таблиці, опис масиву форматів клітин та змінних позиціонування інформації на екрані.

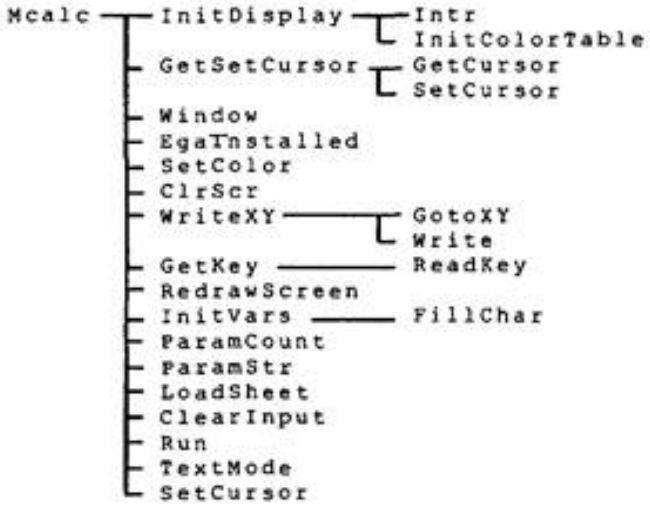
```
Cell : array [1..MAXCOLS, 1..MAXROWS] of CellPtr;
CurCell: CellPtr; {Покажчик на поточну клітинку}
Формат: array [1..MAXCOLS, 1..MAXROWS] of Byte;
LeftCol, RightCol, TopRow, BottomRow,
{Позиціонування}
CurCol, CurRow, LastCol, LastRow: Word;
....
```

Слід зазначити, що виділення окремого масиву форматів інформації клітин не є виправданим. Найпрактичніше ввести байт інформації формату клітини в тип CellRec.

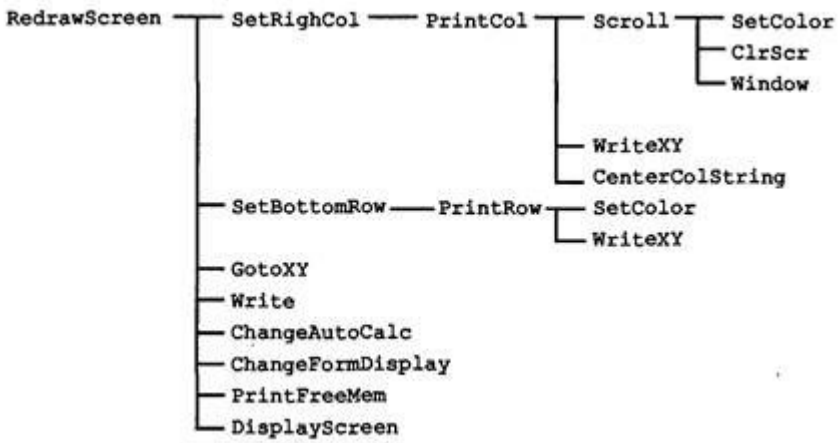
Порівняйте цей проект структури даних із проектом структури даних електронної таблиці, представленої в гол. 3.

Для складання проектної документації, що залишилася, виконаємо трасування програми. Після подвійного натискання клавіші <F7> починає виконуватися налаштування коду, що міститься у файлах *.TPU, і далі починають виконуватися оператори основної програми program Mcalc, що знаходиться у файлі mcalc.pas.

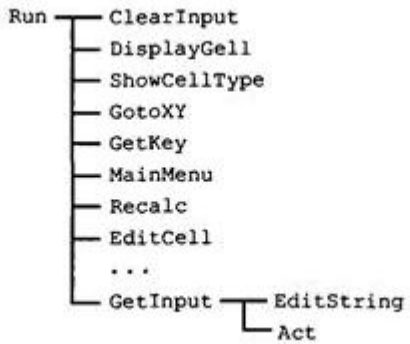
В результаті досліджень було виявлено схему ієрархії модулів програми, зображену на рис. 7.3-7.5. Розшифрування позначень схеми ієрархії представлено у табл. 7.1.



*Мал. 7.3.*Фрагмент схеми ієрархії основних модулів програми



Мал. 7.4. Схема ієрархії модуля RedrawScreen



*Мал. 7.5.*Скорочена схема ієрархії модуля Run

Таблиця 7.1

Розшифрування позначень схеми ієрархії

Ім'я модуля	Файл	Призначення модуля
Act	Mclib	Обробляє інформацію введеного рядка, заносячи її в клітину
CenterColString	Mcutil	Розраховує X координату центрованого в полі виведення рядка

ChangeAutoCalc	Mclib	Встановлює автоматичний та ручний режими рекалькуляції таблиці
ChangeFormDisplay	Mclib	Встановлює режим видимості значень формул або тексту формул
ClearInput	Mcdisplay	Очищає на екрані поле рядка введення
ClrScr	Crt	Очищає інформацію у вікні екрану
DisplayCell	Mclib	Виводить на екран інформацію клітини
DisplayScreen	Mclib	Відображає на екрані внутрішню інформацію таблиці
EditCell	Mcommand	Здійснює редагування вмісту клітини
EditString	Mcinput	Редактор текстового рядка
EgaInstalled	Mcdisplay	Функція, що визначає наявність відеокарти EGA
FillChar	Dos	Надає елементам масиву значення символу
GetCursor	Mcdisplay	Зчитує товщину курсору у змінну
GetInput	Mcinput	Отримавши перший символ, продовжує введення інформації клітини
GetKey	Mcinput	Формує слово розширеного коду клавіші
GetSetCursor	Mcdisplay	Зчитує товщину курсору в змінну та встановлює нову товщину курсору
GotoXY	Mcdisplay	Переміщує курсор відповідно до заданих координат дисплея
InitColorTable	Mcdisplay	Ініціалізує масив перерахунку кольорів для монохромного монітора
InitDisplay	Mcdisplay	Ініціалізує відеокарту на роботу в режимі 80-25
InitVars	Mcutil	Ініціалізує значення основних змінних програми
Intr	Dos	Викликає переривання MS DOS
LoadSheet	Mcommand	Завантажує інформацію таблиці із файлу
MainMenu	Mcommand	Реалізує вибір тем меню програми
Mcalc	Mcalc	Головна програма
ParamCount	Dos	Лічильник полів командного рядка запуску програми Mcalc
ParamStr	Dos	Повертає значення заданого поля командного рядка запуску програми Mcalc
PrintCol	Mcdisplay	Виводить значення координати колонки таблиці
PrintFreeMem	Mcdisplay	Виводить на екран значення залишку вільної пам'яті
PrintRow	Mcdisplay	Виводить значення координати рядка таблиці
ReadKey	Mcinput	Зчитує короткий код однієї клавіші
Recalc	Mclib	Здійснює перерахунок значень формул клітин таблиці
RedrawScreen	Mclib	Відображає на екрані всю інформацію таблиці
Run	Mcalc	Головний цикл програми
Scroll	Mcdisplay	Прокручує інформацію екрана у вказаному напрямку; встановлює колір фону частини екрана, що звільнилася.
SetBottomRow	Mcdisplay	Виводить на екран стовпець із номерами рядків таблиці
SetColor	Mcdisplay	Встановлює колір виведення рядків на екран
SetCursor	Mcdisplay	Встановлює задану товщину курсору
SetRightCol	Mcdisplay	Виводить на екран рядок із найменуваннями стовпців таблиці
ShowCellType	Mcdisplay	Виводить на екран напис про тип поточної клітини таблиці
TextMode	Dos	Переводить екран у вказаний текстовий режим
Window	Crt	Визначає вікно на екрані дисплея
Write	-	Оператор виведення мови Pascal
WriteXY	Mcdisplay	Здійснює виведення заданого числа символів заданого рядка за заданими координатами дисплея

Розглянемо функціональний опис основного ядра програми. У файлі mcutil.pas виконується рудиментарний код, що залишився від колишніх розробок:

```
HeapError := @HeapFunc;
```

У файлі mcdisplay.pas послідовно виконуються підпрограми: InitDisplay, GetSetCursor, Window, EGAInsalled.
Процедура InitDisplay ініціалізує відеокарту на роботу в режимі 80 • 25 за допомогою переривання 10h і викликом процедури InitColorTable ініціалізує масив перерахунку кольорів для монохромного монітора. Останній масив використовується для викликів процедури SetColor.
Процедура GetSetCursor за допомогою процедури GetCursor зчитує товщину курсора змінну OldCursor і за допомогою процедури SetCursor встановлює нову товщину курсора (NOCURSOR).
Процедура Window визначає вікно на екрані дисплея розміщення інформації всієї таблиці. Далі починає виконуватись код головної програми Mcalc.
Наданням CheckBreak := False забороняється використання клавіші <Ctrl+Break> негайного завершення програми.
Виведення початкової заставки здійснюється наступними викликами підпрограм. Процедурами SetColor та ClrScr проводиться очищення вікна програми. Подвійним викликом процедур SetColor та WriteXY виводяться два рядки початкової заставки. Незважаючи на відсутність курсору, відпрацьовується рудиментарний виклик "приховування" курсору GotoXY (80,25). За допомогою функції GetKey здійснюється очікування натискання користувачем будь-якої кнопки.
Процедурами SetColor та ClrScr проводиться очищення вікна програми.
Викликом процедури InitVars ініціалізуються значення основних змінних програм. Масиви ініціалізуються значеннями за промовчанням виклику процедури FillChar.
Присвоюванням Changed := False вказується факт незмінності інформації клітин таблиці після ініціалізації змінних для заборони спрацьовування автозбереження.
Викликом процедури RedrawScreen здійснюється відображення на екрані всієї інформації таблиці.
Якщо значення ParamCount = 1, то командному рядку MS DOS виклику програми було вказано ім'я файлу таблиці. У цьому випадку виконується процедура LoadSheet, яка завантажує інформацію таблиці з файлу з ім'ям файлу, отриманого за допомогою функції ParamStr.
Зрештою, відпрацьовує «зайвий» виклик Clearlnput, який дублюється на початку наступної процедури Run, що містить головний цикл програми.
Після завершення виконання програми послідовно проводиться установка кольору екрана, викликом TextMode переводиться екран у текстовий режим, запам'ятаний змінної OldMode, і, нарешті, викликом SetCursor відновлюється товщина курсора, запам'ятана змінної OldCursor.
Робота процедури RedrawScreen полягає у послідовному виведенні на екран інформації:
• процедурою SetRightCol виводиться на екран рядок із найменуваннями стовпців таблиці;
• процедурою SetBottomRow виводиться на екран колонка із номерами рядків таблиці;
• процедурами GotoXY та Write виводяться написи у верхньому рядку екрана, хоча є зручніша процедура WriteXY;
• виводиться кількість залишку байт пам'яті;
• Процедура DisplayScreen відображає на екрані внутрішню інформацію таблиці.
Зовнішній вигляд програми Mcalc наведено на рис. 7.6.
Робота процедури Run починається із встановлення змінної головного циклу Stop := False та виконання процедури Clearlnput. Головний цикл програми виконується до зміни значення змінної Stop True. Така зміна можлива лише при виборі користувача теми меню Quit — завершення роботи з програмою.
Усередині головного циклу послідовно виконуються такі действия:
- за допомогою процедури DisplayCell виводиться на екран підсвічена клітинним курсором поточна клітина (клітина A1 на рис. 7.6);
- за допомогою процедури ShowCellType виводиться в нижньому лівому куті екрану напис типу поточної клітини таблиці (див. рис. 7.6);
- Оператором Input := GetKey в змінну Input вводиться код символу клавіші, натиснутої користувачем;
— виконуються дії відпрацювання клавіші, натиснутої користувачем.
Дії відпрацювання клавіші, натиснутої користувачем, являють собою ланцюжок альтернативних дій, реалізований структурою ВИБІР. Спочатку відпрацьовуються дії гарячих клавіш. У секції default (якщо

клавіша не була гарячою) викликом процедури Getlnput починається занесення інформації в поточну клітинку таблиці. Процедура Getlnput, занісши символ Input в рядок, що редагується, спочатку викликає EditString — редактор текстового рядка інформації клітини і потім викликає процедуру Act, яка обробляє інформацію введеного рядка, заносячи її в клітинку.

*Мал. 7.6.*Зовнішній вигляд програми Mcalc

Аналіз схеми ієрархії програми та функціонального опису основного ядра програми показав, що основна програма перевантажена допоміжними діями, виділення процедури Run є штучним розподілом основної програми без продуманого структурного розбиття. Усе це призводить до втрати зрозумілості тексту програми.

З метою підвищення зрозумілості програми було ухвалено нові проектні рішення, відображені схемою ієрархії (рис. 7.7).

Виконання основної програми Mcalc починається із запуску нового модуля Starting підготовчих дій програми. Модуль Starting є монітором послідовного виконання модулів InitDisplay, Greeter, InitVars.

Новий модуль InitDisplay тепер є монітором послідовного виконання модулів GetSetMode, GetCursor, SetCursor, Egalnstalled, Window, InitColorTable.

У нового модуля GetSetMode явно як вхідний параметр вказується новий встановлюваний відеорежим, а на виході - старий відеорежим. Така організація краще прямого виклику Intr, оскільки за списком формальних параметрів ясно видно призначення модуля. Реалізація двох функцій виявлення і встановлення відеорежимів в одному модулі тут цілком виправдана, оскільки всі вони реалізуються викликом одного переривання.

Не виправдано використання модуля з двома функціями GetSetCursor, який був монітором послідовного виконання модулів GetCursor, SetCursor. Цей модуль виключено із проекту. Усі функції виведення початкової заставки передані новому модулю Greeter.

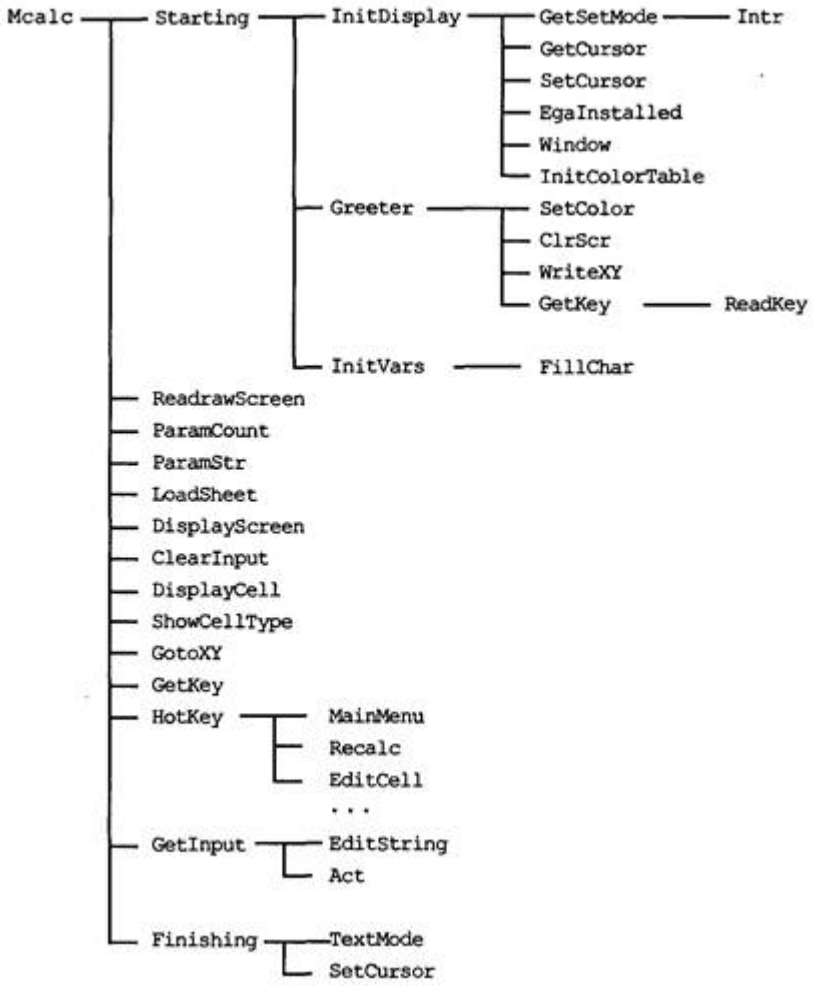
З модуля RedrawScreen виключено виклик модуля DisplayScreen. Це дозволило уникнути повторного виклику модуля DisplayScreen у модулі LoadSheet. Також виправлено помилку використання операторів Write для виведення інформації на екран шляхом використання викликів процедури WriteXY.

Далі починає виконуватися головний цикл програми. Модуль Run вилучений із проекту з метою збільшення зрозумілості програми. Довгий текст вибору дій за кодом натиснутої користувачем клавіші замінений однією альтернативою:

```
If (not(HotKey(Input)) and (ConditionalKey(Input)))
then
GetInput(Input).
```

Нова функція HotKey у разі натискання користувачем гарячої клавіші повертає значення TRUE, інакше функція повертає значення FALSE.

Нова функція ConditionalKey у разі натискання користувачем клавіші з кондиційним для занесення в таблицю кодом повертає значення TRUE, інакше функція повертає значення FALSE.



*Мал. 7.7.*Перероблена схема ієрархії модулів програми

Нова процедура WriteXY тепер не використовує виклик повільної процедури GotoXY і оператор Write, що повільно виконується, і використовує прямий доступ до відеопам'яті. Це дозволило значно прискорити виведення інформації на екран. Більше того, в процедуру доданий новий параметр атрибута кольору рядка, що дозволило уникнути ланцюжків початкового виклику SetColor, а потім WriteXY. Завершується виконання програми викликом нового модуля Finishing. Цей приклад показав самодостатність обраної проектної документації для отримання нового оптимального варіанта побудови структури програми.

ВИСНОВКИ

• Структура програми — штучно виділені програмістом частини програми, що взаємодіють. Використання раціональної структури усуває проблему складності розробки; робить програму зрозумілою людям; підвищує надійність роботи програми при скороченні терміну її тестування та термінів розробки загалом.

• Модуль – функціональний елемент технології структурного програмування. Це підпрограма, але оформлена відповідно до особливих правил.

• Поняття структури програми включає склад і опис зв'язків усіх модулів, які реалізують самостійні функції програми та опис носіїв даних, що беруть участь в обміні як між окремими підпрограмами, так і введеними та виведеними з/на зовнішніх пристроїв.

• Ймовірно, найбільш загальна тактика програмування полягає у розкладанні процесу на окремі дії: функціонального опису на підфункції, а відповідних програм – на окремі інструкції.

• Найголовнішим у схемі ієрархії є мінімізація зусиль зі складання та тестування програми. При використанні заглушок можна добре тестувати сполучення модулів, але не самі модулі. Тестування самих модулів вимагатиме витончених складних заглушок та астрономічного числа тестів. Вихід - до інтеграції модулів тестувати модулі з використанням провідних програм.

• Схема ієрархії повинна відображатися на файлах з вихідними текстами програм таким чином, щоб кожен файл містив якнайбільше готових функцій із загальним призначенням. Це полегшить їх використання у подальших розробках.

Контрольні питання

1. Дайте визначення поняття «структура програми».
2. Що таке модуль програми і які характеристики він повинен мати?
3. Що відбиває схема ієрархії?
4. Яких принципів необхідно дотримуватися, якщо дотримуватися технологій структурного програмування?
5. Дайте визначення поняття «заглушка модуля».
6. Перерахуйте основні засоби зміни топології схеми ієрархії програми.
7. Назвіть критерії оцінки якості схеми ієрархії.
8. Навіщо потрібен паспорт модуля?