Додаток 3 ФОНД ЕВРИСТИЧНИХ ПРИЙОМ ПРОЕКТУВАННЯ ПРОГРАМ

1. ВИБІР СТРАТЕГІЇ ПРОЕКТУВАННЯ ПРОГРАМ
1.1. Замінити висхідний спосіб проектування програм низхідним.
1.2. Інверсія прийому.
1.3. Використовувати комбінований (висхідно-низхідний) спосіб проектування. У разі головна частина програми розробляється низхідним методом, а окремі модулі і підсистеми — висхідним.
1.4. Використовувати спосіб проектування шляхом розширення ядра системи. У разі спочатку створюється оболонка, реалізує мінімальний набір функцій проектованої системи, потім до цієї оболонці (ядру) системи послідовно додаються нові модулі, що розширюють набір функцій, що реалізуються.
2. ВИБІР ПІДХОДУ У ПРОГРАМУВАННІ (методології проектування)
2.1. Замінити методологію, орієнтовану на обробку (модульне програмування; функціональна декомпозиція; проектування з використанням потоку даних; структурне проектування; технологія структурного аналізу проекту SADT; проектування, засноване на використанні структур даних; методологія Джексона; методологія Уорнера та ін.), на методологію, орієнтовану на дані (абстракції даних Дейкстри, об'єктно-орієнтована методологія; методологія, орієнтована проектування концептуальних баз даних та інших.).
2.2. Інверсія прийому.
3. ВИБІР МОВИ
3.1. Вибрати «улюбленішу» мову програмування.
3.2. Вибрати мову програмування, спеціально призначену для вирішення конкретної проблеми.
3.3. Замінити проблемно-орієнтовану мову на об'єктно-орієнтовану.
3.4. Інверсія прийому.
3.5. Замінити мову високого рівня мовою низького рівня.
3.6. Інверсія прийому.
3.7. Використовувати у проекті дві та більше мов програмування.
3.8. Підключати об'єктний код (відкомпільований компілятором іншої мови програмування або асемблер) за допомогою директиви компілятора.
3.9. Використовувати вбудований асемблер системи програмування.
4. ПЕРЕТВОРЕННЯ АРХІТЕКТУРИ, АБО СТРУКТУРИ ПРОГРАМНОЇ СИСТЕМИ
4.1. Збільшити кількість модулів системи.
4.2. Інверсія прийому.
4.3. Замінити глобальну змінну фактичним параметром, що передається модулю як аргумент. Цим прийомом виключається можливість непередбачених змін світових змінних.
4.4. Інверсія прийому.
4.5. Замінити глобальні змінні локальними змінними.
4.6. Інверсія прийому.
4.7. Виконати декомпозицію модуля на кілька. Даний прийом дозволяє розподілити функції, що виконуються між окремими функціями.
4.8. Поєднати кілька модулів в один. Цей прийом дає можливість заощадити час виробництва обчислень; дає особливий ефект, коли дозволяє виключити дублювання тих самих процесів у різних модулях.
4.9. Оформити модулі, пов'язані між собою єдиною логікою, бібліотеку.
4.10. Використовувати у проектуванні системи стандартні модулі системи програмування.
4.11. Використовувати бібліотечні модулі, розроблені іншими програмістами.
5. ПЕРЕТВОРЕННЯ СТРУКТУРИ МОДУЛЯ
5.1. Замінити лінійну структуру команд циклічною. (Підвищує компактність коду програми.)
5.2. Інверсія прийому.
5.3. Замінити гілку структуру циклічної.
5.4. Інверсія прийому.
5.5. Замінити розгалужену структуру if - then - else варіантом оператора case.
5.6. Замінити розгалужену структуру case ланцюжком операторів if - then.
5.7. Інверсія прийому.

5.8. Замінити цикл repeat - until циклом while.
5.9. Інверсія прийому.
5.10. Замінити цикл repeat-until циклом for.
5.11. Інверсія прийому.
5.12. Замінити цикл при циклі for.
5.13. Інверсія прийому.
5.14. Виділити тіло циклу окрему підпрограму. Цей прийом підвищує читабельність програми, але його слід використовувати лише тоді, коли це не порушує внутрішньої логіки циклу.
5.15. Використовувати рекурсію.
5.16. Замінити підпрограму-процедуру підпрограмою-функцією. Даний прийом дозволяє отримати додатковий параметр, який видається підпрограмою (наприклад, код помилки).
5.17. Інверсія прийому. Дозволяє уникнути резервування місця під змінну, яка сприймає значення підпрограми-функції.
5.18. Цілком виключити або мінімізувати використання оператора goto. Покращує структуру програми, її читабельність та логіку.
5.19. Використовувати оператор goto для швидкої передачі керування. Дозволяє швидко без залучення додаткових засобів передавати керування іншим процесом. Слід застосовувати лише у випадках, коли перехід є найбільш лаконічним, простим і ясним засобом.
5.20. Використовувати процедуру exit для виходу із підпрограми. Дозволяє обходитися без оператора goto та без ускладнення логіки підпрограми.
5.21. Використовувати директиву компілятора для безболісного використання процедур як функцій і як процедур.
5.22. Використання процедурного типу даних.
5.23. Використовувати покажчики на процедури та функції.
5.24. Збільшити розмірність масиву.
5.25. Інверсія прийому.
5.26. Використовувати тип даних безліч set замість масивів.
5.27. Інверсія прийому.
5.28. Заміна запису фіксованої довжини записом із варіантом.
5.29. Інверсія прийому.
5.30. Замінити звичайні рядки (тип String) рядками із нульовим закінченням.
5.31. Інверсія прийому.
5.32. Використовувати оператор with для спрощення роботи із записами.
5.33. Використовувати перетворення типів даних.
5.34. Використовувати типізовані константи.
5.35. Давати змінним, константам та типам даних змістовні позначення.
5.36. Широко використовувати коментарі пояснення обчислювальних алгоритмів.
6. ОРГАНІЗАЦІЯ І ЗБЕРІГАННЯ ДАНИХ
6.1. Замінити типізований файл на нетипізований файл.
6.2. Інверсія прийому.
6.3. Замінити типізований файл текстовим файлом.
6.4. Інверсія прийому.
6.5. Замінити нетипізований файл текстовим файлом.
6.6. Інверсія прийому.
6.7. Замінити носій даних.
6.8. Проводити сортування даних з метою полегшення пошуку.
6.9. Використовувати індексовані масиви даних для організації пошуку за вторинними ключами.
6.10. Виключити надмірність даних.
6.11. Декомпозувати дані на кілька файлів.
6.12. Об'єднати дані в один файл даних.
7. ЕКОНОМІЯ РЕСУРСІВ ПРОГРАМИ
7.1. Використовувати inline-процедури та inline-директиви. Дозволяє економити пам'ять комп'ютера та збільшує швидкодію алгоритму, оскільки реалізація такого ж алгоритму за допомогою операторів мови високого рівня після компіляції призводить до збільшення об'єктного коду та ускладнення алгоритму за

рахунок додавання різних операторів контролю кордонів тощо. У процедурах inline здійснюється безпосереднє введення тексту у машинних кодах, і вся відповідальність щодо організації процесу лежить на програмісті.
7.2. Використовувати директиви вбудованого асемблера.
7.3. Використовувати абсолютну адресацію даних через директиву absolute та стандартні масиви Mem, MemW, MemL.
7.4. Використовувати безпосереднє звернення до портів через стандартні масиви Port, PortW, PortL.
7.5. Використовувати систему переривань через функції модуля DOS - Intr та MS DOS.
7.6. Використовувати профіль коду програм за допомогою програм-профілювальників.
7.7. Замінити статичні змінні та масиви динамічними.
7.8. Використовувати оверлейну організацію програм.
7.9. Об'єднати оверлейні файли в один файл типу *.EXE.
7.10. Розбити програму на резидентну частину (TSR) і частини, що підвантажуються.
7.11. Використовуйте додаткову пам'ять комп'ютера (expanded memory).
7.12. використовувати розширену пам'ять комп'ютера (extended memory).
7.13. Використовувати захищений режим процесора (protected mode).
7.14. Використання режиму віртуального процесора 8086.
8. ОФОРМЛЕННЯ ВАРІАНТУ (ВЕРСІЇ) ПРОГРАМИ
8.1. Розмноження околиці (копіювання старого варіанту окремий файл). Вкрай неефективний метод через захаращення дискового простору.
8.2. Заміна виклику старої процедури на виклик нової також неефективна, оскільки старі процедури також підключаються до об'єктного коду програми, що призводить до захаращування програми.
8.3. Використання оператора вибору. Ті самі обмеження.
8.4. Коментування зміненого коду програми.
8.5. Використання директив компілятора {$IFDEF <умова>} та {$IFOPT <опція>}.
9. ТЕСТУВАННЯ ПРОГРАМ
9.1. Замінити висхідне проектування тестів низхідним.
9.2. Інверсія прийому.
9.3. Використовувати метод великого стрибка.
9.4. Використовувати метод «сандвіч».
9.5. Організувати вхідні дані для тестування у зовнішньому файлі. Це виключить повторне введення даних при кожному тестуванні, що дозволить заощадити час.
9.6. Використовуйте генератор вхідних даних.
10. НАЛАДКА ПРОГРАМ
10.1. Використовувати вбудований налагоджувач системи (трасування програми).
10.2. Використовувати директиви компілятора {$D} та {$L} при компіляції модулів з метою мати безпосередній доступ до змінних та процедур модуля.
10.3. Використовувати налагоджувальний друк. Виводити значення окремих ключових змінних і масивів безпосередньо на екран або зовнішній файл на диску.
10.4. Вставити «заглушки» на ті модулі програми, які зараз не налагоджуються.
10.5. Використовувати процедуру halt у разі виняткової ситуації.
10.6. Використовувати повернення функцією або процедурою спеціального значення у разі виняткової ситуації.
10.7. Використовувати код повернення у вигляді окремої глобальної змінної.
11. ОРГАНІЗАЦІЯ ДІАЛОГУ З КОРИСТУВАЧЕМ
11.1. Замінити горизонтальне меню вертикальним меню.
11.2. Інверсія прийому.
11.3. Використовувати скролінг меню.
11.4. Замінити спливаюче меню, що випадає.
11.5. Інверсія прийому.
11.6. Організувати меню, яке активується за гарячими клавішами.
11.7. Використовувати кнопки та панелі діалогу.
11.8. Організовувати громіздкі екранні форми як багатосторінкових форм.
11.9. Використовувати скролінг екранних форм.

11.10. Використовувати спливаючі екранні форми.
11.11. Використовувати гіпертекстову систему як систему допомоги.