**Реферат на тему:**

**Структурне програмування**

**План**

1. Структурне програмування
2. Принцип модульності
3. Процедурна абстракція. Модулі в Turbo Pascal.
4. Література

**1. Структурне програмування**

За часів стихійного програмування хорошими програмістами вважали тих, хто створював досить хитромудрі програми, які займали мінімум часу та пам’яті при виконанні. Це було цілком природно, враховуючи тодішні можливості обчислювальної техніки. Результатом такого програмування виявлялись програми, які було важко (якщо взагалі можливо) зрозуміти іншим. Навіть автори таких програм з часом з трудом розуміли власне творіння. Внесення необхідних змін в таку програму робило ситуацію ще більш заплутаною. Подібні програми одержали назву BS-програм (це абревіатура від “bowl of spaghetti” – блюдо спагетті, бо саме так виглядала програма при спробі зобразити всі переходи між її операторами) [15]. Піонер структурного програмування Е. Дейкстра навіть проголосив, що “кваліфікація програміста обернено пропорційна кількості операторів безумовного переходу в його програмах”. Структурне програмування іноді називають “програмування без go to”, хоча це екстремальна точка зору. Насправді мова йде про те, щоб не використовувати оператори переходу без особливої необхідності. Перш за все структурне програмування мало своєю метою позбавитись від поганої структури в програмі. Ще однією метою було створення таких програм, які були б легко зрозумілими навіть без їх авторів, адже “*програми пишуться для людей* – *комп’ютером вони лише обробляються*”. Зміст цієї фрази полягає у тому, що трансляція і виконання програми будь-якої структури на комп’ютері дійсно не викликає ніяких труднощів. А от роботу по перевірці правильності програми, внесення виправлень і змін доводиться виконувати людині.

Отже, ***структурне програмування*** є технологією програмування, яка об’єднує способи складання добре структурованих надійних програм, зручних для читання і розуміння їх людиною, слідкування за логікою їх роботи, внесення до них виправлень та інших змін. Згідно з думкою Н.Вірта “структурізація є принциповим інструментом, яке допомагає програмісту систематично синтезувати складні програми, зберігаючи про них повне уявлення” [1].

Реалізація цих ідей заснована на таких ***принципах***:

1. *аналітичне* (згори донизу) *програмування*;
2. *структурне кодування , тобто використання лише базових елементів програми*;
3. *принцип модульності*.

З точки зору структурного програмування, ***правильна програма*** – це програма, структура якої включає тільки базові елементи, і жоден з цих базових елементів не є недоступним і не допускає зациклювання. Правильна програма має тільки один вхід і тільки один вихід. В правильній програмі не повинно бути таких частин, які ніколи не виконуються.

# *1.1. Принцип модульності*

*Принцип модульності* полягає у тому, що програма розбивається на логічно незалежні частини (модулі), які дотримуються зв’язків. Історично поняття модульної програми виникло раніше, ніж були сформульовані принципи структурного програмування, проте ця ідея виявилася просто необхідною складовою нової технології програмування разом з аналітичним проектуванням.

Поняття модуля цілком логічно з’являється на відповідному етапі аналітичного програмування: *модуль* – це частина програми, яка розв’язує порівняно нескладну задачу, логічно незалежну від інших задач.

Зовсім просто: модулі – це підпрограми (процедури або функції), які мають певні властивості.

*Деякі необхідні властивості модуля:*

1. єдиний вхід, єдиний вихід (деякі мови дозволяють існування декількох входів або виходів);
2. окрема компіляція;
3. кожний модуль доступний за своїм ідентифікатором;
4. модуль може викликати інший модуль;
5. модуль не повинен зберігати історію своїх викликів (інакше може виникати так званий *побічний ефект*);
6. модуль порівняно невеликий;
7. кожен модуль відповідає лише одній задачі;
8. незалежність функціонування (заміна модуля на аналогічний не впливає на всю програму).

З часом, коли принцип модульності став підтримуватись мовами програмування, на перший план висунулась вимога логічної та програмної незалежності модуля.

Слід зауважити, що *повної незалежності між модулями бути не може*. Залежність між модулями існує:

* через списки параметрів;
* тоді, коли вони користуються спільними (глобальними) змінними;
* всі модулі програми залежать від структури даних цієї програми;
* модулі залежать від логіки функціонування програми (від того фактору, що модуль може викликатися іншими модулями).

Отже, *модулі повинні бути незалежні в межах інтерфейсу програми і структури даних*. Практика показала, що чим вищий степінь незалежності модулів, тим простіше розібратись в окремих модулях і в програмі в цілому; тим менша ймовірність з’явлення нових помилок при виправленні старих, або внесенні змін в програму, тобто менша ймовірність так званого ***хвильового ефекту***.

Із сказаного вище випливає, що не слід без крайньої необхідності використовувати в модулях глобальні змінні. Всі зв’язки між модулями повинні підтримуватися через списки параметрів.

# *1.2. Процедурна абстракція. Модулі в Turbo Pascal.*

*Процедурна абстракція* – це філософія, яка полягає у тому, що при розробці підпрограм піклування про те, ***що*** повинна виконувати процедура або функція, відокремлюється від того, ***як*** це виконується.

Потенціал процедурної абстракції більш повно можна реалізувати не у стандартній реалізації мови Паскаль, а у Turbo Pascal (TP).

Після утворення якоїсь процедури, яка має універсальне значення, TP надає вам можливість помістити її у свою особисту бібліотеку підпрограм, або *модуль* (unit). Потім цей модуль можна використати (імпортувати цю процедуру) в іншій програмі. При цьому ви дійсно відокремлююте ***що*** від ***як***. Знаючи, що робить дана процедура і як її викликати, ви можете без обмежень її використовувати, не маючи при цьому жодного уявлення, як саме вона реалізована [7, 11].

Модулі TP знаходяться в окремих файлах на диску. Вони можуть містити описи констант, типів даних, змінних, процедур і функцій. Компілювати модуль і виправляти у ньому знайдені помилки можна окремо від програм, у яких цей модуль використовується.

Виклик: uses MyUnit;

Структура модуля аналогічна до структури Паскаль-програми:

unit <ім’я модуля>;

interface

загальнодоступні описи

implementation

приховані описи (змінні, вкладені процедури і т.д.)

описи загальнодоступних процедур і функцій

[розділ ініціалізації]

end.

Розділ interface містить всю інформацію, яку необхідно знати програмісту, щоб користуватися цим модулем. Тут є заголовки і документація (коментарі) всіх підпрограм модуля.

Тіла підпрограм міститься в розділі implementation. Кінець модуля позначається словом end. Для нього не існує відповідного begin, якщо у розділі не існує розділа ініціалізації, у якому змінним, що використовуються всередині модуля, надаються початкові значення. Розділ ініціалізації виконується до виконання будь-якого клієнта модуля (програми або модуля, який використовує наш модуль).

**ЛІТЕРАТУРА**

1. Н. Вирт. Систематическое программирование. – М.: Мир, 1977. – 183 с.
2. Ален И. Голуб. С и С++. Правила программирования. – М.: БИНОМ, 1996. – 272 с.
3. У. Дал, Э. Дейкстра, К. Хоор. Структурное программирование. – М.: Мир, 1973. – 247 с.
4. Э. Дейкстра. Дисциплина программирования. – М.: Мир, 1978. – 275 с.