УВАЖАЕМЫЕ СТУДЕНТЫ! Законспектируйте в своей рабочей тетради по дисциплине приведенную лекцию (объемом 4-5 страницы), ответьте *письменно* на контрольные вопросы.

Результаты работы, фотоотчет, предоставить преподавателю на e-mail: r.bigangel@gmail.com до 23.01.2023.

При возникновении вопросов по приведенному материалу обращаться по следующему номеру телефона: (072)111-37-59, (Viber, WhatsApp), vk.com: https://vk.com/daykini

ВНИМАНИЕ!!! При отправке работы, не забывайте указывать ФИО студента, наименование дисциплины, дата проведения занятия (по расписанию).

Лекция № 29
Тема «Требования к стилю написания программы»

План лекции:

Стиль программирования- набор приемов или методов программирования, которые используют опытные программисты, чтобы получить надежные, эффективные, удобные для применения и легко читаемые программы. Правила хорошего стиля программирования - это результат соглашения между опытными программистами. Когда программист усвоит определенный стиль, его программа станет значительно легче для восприятия. Программа - это документ для последующего использования, учебный материал по кодированию алгоритмов и средство для дальнейшей разработки более совершенных программ. Следовательно, языки программирования должны обеспечивать возможность создания удобочитаемых совокупностей операторов. Язык Турбо Паскаль имеет такие средства обеспечения удобочитаемости.

Программисты должны всегда уметь прочитать свои программы. В этом им должна помогать стандартизация стиля. Хотя нет государственных

стандартов стиля программирования, однако каждая программистская фирма имеет свои правила стиля.

Трудно читаемые программы обычно сложно модифицировать, особенно если это предстоит выполнять не самому автору программы.

Иногда легче полностью переписать чужую программу, чем ее модифицировать.

Следуя определенному рациональному стилю программирования, можно избежать многих трудностей, возникающих при разработке и модификации программ. Легко читаемые программы хорошо передают логику и структуру алгоритма.

Дадим общие рекомендации. Они просты, не требуют для выполнения особых усилий и направлены на более четкую структуризацию программы. Суть их в следующем:

1. При написании программы не следует создавать большие программные модули. Логически завершенные последовательности операторов целесообразно оформлять в виде подпрограмм. Отлаживая их отдельно, легче локализовать и исправить ошибки.

2. Не следует создавать большие текстовые файлы. Подпрограммы лучше компоновать в отдельные модули, размещая их в отдельные файлы. В этом случае при изменении какого-то одного модуля не нужно будет перекомпилировать все остальные.

3. При написании программы целесообразно использовать систему отступов, когда операторы, вложенные в другие операторы или в операторные скобки, пишутся в строке с отступом вправо-влево по отношению к другим операторам (обычно отступ делается в две-три позиции). Рекомендации здесь такие:

- зарезервированные слова типов данных (TYPE, VAR, CONST и т.д.) записывайте в первой позиции строки, а описываемые переменные - в следующей строке с отступом на несколько позиций;

- слова операторных скобок Begin-End, обозначающих соответствующий составной оператор, записывайте в одних и тех же позициях;

- операторы тела цикла сдвигайте правее относительно оператора цикла. Операторы внутренних циклов сдвигайте правее относительно операторов внешних циклов;

- в операторах условной передачи управления IF - THEN - ELSE слова THEN и ELSE, относящиеся к одному условию, целесообразно записывать одно под другим;

- метки лучше всего располагать в самых левых позициях так, чтобы они “не загораживались” другими операторами.

Такое расположение операторов позволяет разобраться со структурой программы, понять ее содержание, быстрее найти некоторые ошибки (например, отсутствие закрывающей операторной скобки END, соответствующей открывающей скобке BEGIN). Примером такого расположения операторов являются программы и их фрагменты, приведенные в данном рабочем учебнике.

4. Не следует в одной строке объединять несколько операторов, за исключением простейших (A:=0; B:=A; P:=0; и т.п.), так как это затруднит локализацию ошибки во время отладки программы, поскольку минимальный выполняемый блок команд в процессе отладки соответствует одной строке текста.

5. Делайте пробелы для улучшения читаемости программы. Широкое использование пробелов существенно облегчает чтение программы. Ставьте пробелы между элементами списка данных, а также до и после операций +, -, =. Иногда желательно отделять пробелами операции (*, /). Пробелы можно также использовать для указания приоритета операций. Например, запись вида I + A*B предпочтительнее, чем I+A * B.

6. Идентификаторы программы должны быть выразительными, т.е. обозначать сущность или сокращенное наименование параметров задачи.

При написании идентификаторов следует широко использовать как строчные, так и прописные буквы, а также знак подчеркивания: MyProgram, File_of_Digits, Volumel, Razmer24 и т.д.

Не опасайтесь использовать длинные имена, т.к. через промежуточный буфер можно скопировать их необходимое число раз.

Правильный выбор имен переменных - это залог удобочитаемости программ. Кроме того, это самый легкий и дешевый метод, так как он требует незначительных умственных усилий от программиста и столь же небольшого расхода ресурсов компьютера. Не используйте схожих по виду имен. Правильно выбранные имена переменных уменьшают необходимость комментариев.

7. Рекомендуется широко использовать написание комментариев, по крайней мере для связанных по смыслу групп операторов или даже отдельных операторов, если необходимо подчеркнуть их особенности. Каждая процедура, функция должна иметь комментарии, в которых сообщается ее назначение.

8. Следует использовать при написании программы возможность расцвечивания разными цветами различных ее элементов, что позволяет делать версия Турбо-Паскаль v.7.0. При этом проще проконтролировать правильность использования зарезервированных слов языка, комментариев, вставок на ассемблере и т.д.

9. Правильное использование скобок существенно улучшает читаемость программы. Так как последовательность выполнения операций определяется их приоритетами, то программист может ограничиться небольшим количеством скобок, но при этом затрудняется чтение и корректировка программы.

Малое количество скобок Дополнительные скобки

A*B*C/(D*E*F) (A*B*C)/(D*E*F)

A*B/C*D/E*F (A*B*D*F)/(C*E)

A/B/C/D ((A/B)/C)/D

Основное правило такое: в сомнительных случаях всегда ставить скобки. Это не только сделает программу более понятной, но и предотвратит ошибки.

10. Во избежание неоднозначностей не следует локальным и глобальным параметрам давать одинаковые имена.

11. Пропуск строк - очень эффективный метод повышения наглядности программы. Последовательности операторов, выполняющих какое-то законченное действие, целесообразно отделять от предыдущих и последующих операторов пустыми строками. Пропуском строк можно разделить программу на отдельные фрагменты. Пропуском одной строки можно отделять каждую группу логически связанных операторов, пропуском двух строк отделяются основные логические фрагменты программы. Использование незаполненных строк облегчает поиск отдельных частей в программе. Пропуск строк до и после комментариев помогает выделить последние.