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 № 24

Тема «Стиль программирования, языки программирования»

План:

1. Стили программирования

2. Языки программирования

Понятие стиль программирования представляет собой совокупность правил, которые необходимо соблюдать при написании программы. Необходимо помнить, что программа написана в "хорошем стиле" программирования, если она написана структурно, легко читаема, в ней корректно используются все "ресурсы" языка программирования и программа сопровождается полным набором тестов.

Правильно разработанные программы должны не только удовлетворять своим функциональным требованиям, но и обладать такими свойствами, как:

- повторная используемость;
- расширяемость;
- устойчивость к неправильным данным;
- системность.

Правильный стиль программирования обеспечивает наличие этих свойств.

Стиль программирования

Этап проектирования программ оказывает влияние на стиль программирования, надежность, эффективность, отладку, тестирование и эксплуатационное свойство программ. Таким образом, это важнейшая часть любой программной разработки.

Работая над программой, программист, особенно начинающий, должен хорошо представлять, что программа, которую он разрабатывает, предназначена, с одной стороны, для пользователя, с другой — для самого программиста. Текст программы нужен прежде всего самому программисту, а также другим людям, с которыми он совместно работает над проектом. Поэтому для того, чтобы работа была эффективной, программа должна быть легко читаемой, ее структура должна соответствовать структуре и алгоритму решаемой задачи. Как этого добиться? Надо следовать правилам хорошего стиля программирования.

Под стилем программирования понимается внутренне согласованная совокупность базовых конструкций программ и способов их композиции, обладающая общими фундаментальными особенностями, как логическими, так и алгоритмическими. Стиль включает также совокупность базовых концепций, связанных с этими программами.

При оформлении текста программы хороший стиль программирования предполагает:

- использование комментариев;
- использование несущих смысловую нагрузку имен переменных, процедур и функций;
- использование отступов;
- использование пустых строк.

Следование правилам хорошего стиля программирования значительно уменьшает вероятность появления ошибок на этапе набора текста, делает

программу легко читаемой, что, в свою очередь, облегчает процессы отладки и внесения изменений.

Четкого критерия оценки степени соответствия программы хорошему стилю программирования не существует. Вместе с тем достаточно одного взгляда, чтобы понять, соответствует программа хорошему стилю или нет.

Сводить понятие стиля программирования только к правилам записи текста программы было бы неверно. Стиль, которого придерживается программист, проявляется во время работы программы.

Очевидно, что хороший программист должен следовать правилам хорошего стиля:

1. *Стремление к простоте*. Простота проектирования программ – это первый шаг, ведущий к получению легко читаемой программы. Необычное кодирование программы (например, использование скрытых возможностей машины) часто препятствует отладке программы и конечно затрудняет ее использование другими программистами. Структура программы должна раскрывать ее логику.

2. *Чтение программы*. Программу надо снабжать комментариями и параграфами.

3. *Описание задач*. Идеальная программа дает точный желаемый выход при неясно сформированных требованиях пользователя.

4. *Постановка задачи*. В данном разделе надо четко определить задачу, ее цели, этапы и конечный результат. Далее программист должен переписать спецификации задачи ориентируясь на ЭВМ. Необходимо сжатое, но полное описание программы. Затем программист и заказчик должны тщательно изучить написанные спецификации задачи, чтобы быть уверенным в ее правильном понимании.

5. *Выбор алгоритма*. Важнейшим шагом для получения эффективной и правильной программы является составление алгоритма. При этом предполагается правильный выбор языка и спецификации программы. Таким образом, хороший алгоритм необходимое, но недостаточное условие

хорошей программы. Если пользователь формирует задачу в виде четкого алгоритма, то процесс проектирования существенно облегчается. Для эффективности необходимо рассмотреть несколько алгоритмов и из них выбрать наиболее эффективный.

6. *Описание данных.* Другим фактором сравнимым по значению с выбором алгоритма является описание данных. Хорошо продуманное описание данных существенно сокращает программу. Так, полезно, использовать массивы данных в том случае, когда это наиболее очевидный способ их организации. Другой пример такого рода – возможность использовать ссылки и указатели. Например, если нужно проследить отношения между родителями и их потомками на протяжении нескольких поколений, то легче всего это сделать с помощью ссылок и указателей.

7. *Выбор языка программирования.* Часто выбор языка программирования предоставлен данной выигрышной системе, или подготовкой программиста. Существуют серьезные основания для установления языковых стандартов для системы. Если применяют много разных языков для написания программ, то использование последних становится затруднительным.

8. *Универсальность.* Хорошая универсальная программа должна обрабатывать вырожденные случаи (например, число элементов равно 0 ил 1) и печатать сообщения об ошибке. Тогда программа является не только универсальной, но и защищенной от ошибок. Используйте в качестве компиляторов переменные, а не константы.

9. *Библиотеки.* Чтобы повысить эффективность разработки программ, облегчить отладку и тестирование, а следовательно и сократить работу по созданию программ используйте библиотеки. Тип библиотечных подпрограмм представляет собой функции и подпрограммы имеющиеся в наличии для данного языка программирования.

10. *Форматы ввода/вывода.* Форматы входных и выходных данных являются частью этапа проектирования, входные данные должны быть

разработаны с учетом максимального удобства для пользователя и минимальные ошибки. Постоянство входных форматов, как правило, также способствуют уменьшению ошибок. Выходные спецификации могут сильно различаться. Иногда даются четкие инструкции и выходные данные подгоняются под определенный стандарт. Однако часто отсутствуют какие-либо указания и выходные данные подчас представляют собой страницы, заполненные числами без всякой идентификации. Выходная информация должна идентифицироваться без привлечения других источников. Выходные данные должны содержать: 1. идентификацию выходной информации, 2. описание записи, 3. дату, 4. нумерацию страниц. Кроме того, каждая напечатанная группа элементов должна быть помечена. При табличной форме выдачи должны быть помечены строки и столбцы.

Заметим, что в программировании происходит систематический пересмотр понятий. То, что в других местах называется орудием, здесь называется методом; то, что называется методом или методикой, называется методологией; то, что называется методологией, возводится в ранг парадигмы, и т. д. Поэтому можно выделить, в частности, следующие стили программирования:

– Программирование от состояний.

– Структурное программирование.

– Сентенциальное программирование.

– Программирование от событий.

– Программирование от процессов и приоритетов.

– С программистской точки зрения эти стили различаются по следующим характеристикам.

– Структурное программирование, рассматривают как монопольный первоуровневый стиль.

Другие перечисленные стили программирования являются предметом изучения в ВУЗах. Отметим лишь, что для описания их принципов используются такие математические категории, как функционал, предикаты

и др. Из этого следует, что математический анализ годится для описания физических, но не годится для описания информационных процессов. Для информатиков намного важнее логика, алгебра, топология, лингвистика и философия. Для них важнее преобразовывать понятия, чем формулы.

Языки программирования

Использование ЭВМ немыслимо без программирования, которое в самом простом понимании представляет собой создание программ. Более точно, программирование заключается в отображении в памяти ЭВМ цифровых данных о реальных объектах и в описании на машинном языке инструкций по управлению этими данными. Так как для восприятия человеком машинные языки неудобны, то для более эффективной работы были созданы различные языки программирования.

Машинный язык относится к внутреннему уровню представления обработки данных. Человек обращается к машинному языку в очень специфических случаях.

Алгоритмические языки представляют собой внешний уровень и имеют форму, удобную для восприятия человеком.

Наиболее распространены так называемые процедурные языки программирования. Процедурный язык программирования предоставляет набор типов и операций с этими типами, а также средства для логической организации программы. Программа на процедурном языке выполняется поэтапно - оператор за оператором. Наиболее распространёнными процедурными языками программирования являются: C, C++, Fortran, Pascal, Basic, Visual Basic, Ada.

Язык программирования Fortran был создан в 1956 г. и до 70-х годов использовался в подавляющем числе программных проектов. На сегодняшний день имеется огромное число прикладных программ, созданных на этом языке, поэтому практическое использование Fortran-а продолжается. Однако область использования этого языка программирования ограничена численными расчётами в области физики. Все суперЭВМ имеют

в составе своего программного обеспечения средства для работы с Fortran-ом. Для Fortran-а имеется международный стандарт, что позволяет создавать хорошо переносимые программы.

Язык Basic представлял собой упрощённый Fortran и был создан в 1964 г. для начального обучения программированию. Однако со временем этот язык приобрёл популярность среди профессиональных разработчиков программ. Основной недостаток языка – это использования большого числа «правил по умолчанию», что затрудняет создание надёжных программ. Отсутствие общепринятого стандарта на язык также мешает его распространению и использованию в серьёзных программных разработках. Язык Visual Basic является объектно-ориентированной версией языка Basic, созданной фирмой Microsoft, и широко используется для разработки графического интерфейса прикладных программ. Он появился в 1991 году, но своими корнями уходит к программе Ruby, написанной Аланом Купером (Alan Cooper) в 1988 году.

Язык Pascal был создан в 1970 г. также для изучения программирования, однако, многочисленные положительные качества этого языка обусловили его широкое распространение как среди прикладных, так и среди системных программистов. Наибольшую популярность приобрела объектно-ориентированная версия этого языка, реализованная фирмой Borland в своей RAD- системе Delphi.

Язык C, созданный в 1972 г. получил распространение как язык системного программирования. На этом языке написаны операционные системы Unix и её многочисленные версии (Linux, IRIX, AIX), а также операционная система Windows NT.

Язык C++, созданный в 1982 г., являясь принципиально иным языком, тем не менее сохранил совместимость с C, а, следовательно, возможность использования ранее созданных программ. Де-факто язык C++ стал стандартом для создания сложных программ как системных, так и прикладных. Язык программирования Ada был разработан в 1979 г. по заказу

Министерства обороны США и является обязательным для многих военных приложений. Основное назначение языка – создание больших программ для работы в реальном времени. Существенным недостатком языка является его громоздкость.

Из непроцедурных языков наиболее известными являются LISP и PROLOG. Язык LISP создан в 1959 г. и рассматривается как основной язык программирования систем искусственного интеллекта.

Язык логического программирования PROLOG создан в 1978 г. и используется для работы с базами знаний, основанными на фактах и правилах. То есть программы, написанные на этом языке, должны обладать некоторой степенью «интеллектуальности».

В зависимости от того, насколько детально учитываются особенности ЭВМ в конкретном языке программирования, говорят об уровне программирования. Язык С является низкоуровневым языком, так как может работать непосредственно с физическими адресами памяти ЭВМ. Языка Ada, LISP, C++ являются высокоуровневыми языками.

Контрольные вопросы:

1. Стили программирования (описать, привести примеры)

2. Языки программирования (описать, привести примеры)

3. Составить сводную таблицу представленных языков программирования