Стиль программирования

**1. Стиль программирования**.

Основной принцип хорошего стиля программирования - писать так, чтобы программу могли прочесть не только машины, но и люди.

**Стиль** - набор приемов или методов программирования, используемых получения правильных, удобных для применения, легко читаемых и, желательно эффективных программ.

**Особый стиль** - это программы, понятные лишь вам.

Приемы формирования листинга просты.

Исторически **листингом** называли распечатку текста программы. Сейчас под этим можно понимать размещение текста программы на любом носителе: бумаге, экране и т.п. Следует записывать 1 инструкцию (кроме сложных) в строку.

**Отступ** - это абзац, выявление структуры программы. Используется, как правило, при записи сложных инструкций, имеющих в своем составе другие инструкции. Рекомендуемый размер отступа для вложенных инструкций 4-5 позиций. Размер отступа должен быть одинаков по всей программе.

**Комментарии**. Цель - пояснить логику, облегчить отладку, тестирование и сопровождение программ. При написании комментариев следует учитывать уровень возможных чита­телей текста программы. Второй момент: когда писать? Рекомендуется записывать комментарии одновременно с текстом программы. После написания текста про­цедуры необходимо прочесть его и во всех местах, где возможен вопрос, проком­ментировать.

Вопрос о количестве (объеме) комментариев является дискуссионным. Их количество зависит от сложности логики программы, числа процедур и используе­мых библиотечных функций, а также от того, является ли программа коллективной разработкой или нет. Можно указать такой косвенный критерий достаточности объема комментирования: объем комментариев должен быть таков, чтобы при чтении ее текста, скажем через **1** год, можно было достаточно легко разобраться в логике и структуре программы. Можно также привести грубый количественный критерий: ориентировочный объем комментариев - **50-70**% от объема кода программы.

Различают следующие типы комментариев:

**- оглавления,**

**- вводные,**

**- поясни­тельные**.

**Оглавления**- Разумно составлять для программ, объем кода которых пре­вышает 300 - 500 КБ.

**Вводные**- записываются перед началом текста процедуры.

**Пояснительные** - служат для пояснения.

**Принцип комментирования**: программа должна быть понятна без привлечения дополнительной документации.

**Имена**. Имена объектов программы должны быть осмысленными. Пример. **tip  razmer  adress**. В настоящее время во всем мире приняты 2 системы записи (нотации) "**паскалевская**" и "**венгерская**". Обе рекомендуют использовать имена, состоящие в  общем случае, из нескольких слов.

В "**паскалевской**" нотации каждое слово начинают с прописной буквы.

Пример. ВедКт  EndKm   FieidName

В "венгерской" нотации перед именем записывается тип объекта программы

Пример. intBegKm   intEndKm   charFieldName

Не используйте в программе сходные имена!

Пример. value   values ах1   axi к  К

Если имя содержит цифры, записывайте их в конце.

Пример. Fund1   Таb15

Имя не должно совпадать с ключевым словом (для языков, где ключевые слова не резервируются) или с именем библиотечной или встроенной функции.

**Общие приемы программирования**

1) Изучайте и используйте библиотечные функции.

2) Не игнорируйте предупреждения и старайтесь их ликвидировать.

3) Старайтесь избегать промежуточных переменных. Формулы следует записывать в том же виде как это принято в математике.

4) Не используйте инструкции переход типа goto. Эти инструкции нарушают естественный порядок выполнения программы. Придерживайтесь принципа: текст программы надо читать как книгу слева направо и сверху вниз.

5) Объявляйте все переменные явно.

6) Не используйте одни и те же переменные для различных целей. Т.к. при последующей модификации одного фрагмента кода ошибки могут возникнуть в другом фрагменте.

7) Активно используйте удобные и понятные особенности языка.

8) Используйте исчерпывающие проверки.

Стиль программирования связан с удобочитаемостью программы.

Правила хорошего стиля программирования – это результат соглашения между опытными программистами.

Правило стандартизации стиля заключается в следующем: если существует более одного способа сделать что-либо и выбор произвольный, остановитесь на одном способе, и всегда его придерживайтесь. Программное средство представленное в хорошем стиле имеет комментарии (пояснительные, вводные иногда оглавления), значимые идентификаторы, хорошо воспринимаемый текст ПС.

Пользовательский интерфейс также должен быть разработан в хорошем стиле, придерживаясь следующих рекомендаций:

* пользовательский интерфейс должен базироваться на терминах и понятиях, знакомых пользователю;
* пользовательский интерфейс должен быть единообразным;
* пользовательский интерфейс должен позволять пользователю исправлять собственные ошибки;
* пользовательский интерфейс должен позволять получение пользователем справочной информации: как по его запросу, так и генерируемой ПС.

Следование приведенным правилам позволяет коды легко читать, отлаживать и редактировать

Лесенка. В основе нотации лежит традиционное правило лесенки. Лесенка (ступенчатые отступы) отражает вложенность команд друг в друга и позволяет легко видеть в тексте границы больших многостроковых конструкций.

Выделение скелета. HTML-теги как скелет пронизывают информационные блоки, предназначенные для вывода на экран. Для выделения тегового каркаса предлагается записывать названия команд заглавными буквами. Эта рекомендация основана на разумном предположении о “мелкосимвольности” информационных блоков. Вместе с тем, предлагается записывать заглавными буквами только названия команд, а имена атрибутов и их значения — малыми. Чтобы не переборщить. Ведь выделить много — значит, не выделить ничего.

Упирание коленок в стену. Информационный блок размещается с первой позиции, а после него оставляется пустая строка. Это правило особенно полезно для текстовых абзацев, ведь создавать и редактировать текст гораздо удобнее, когда он начинается с левого края экрана. Пустая строка после абзаца позволяет выполнять автоматическое переформатирование текста в текстовом редакторе без “поломки” скелета.

Короткие строки. код не должен иметь строк, выходящих за границу экрана текстового редактора. Видеть всю строку на экране — большое удовольствие, а горизонтальная линейка протяжки, наоборот, доводит до белого каления.

Необходимость комментариев.

**Классификация стилей программирования**

Одним из важнейших признаков классификации языков программирования является принадлежность их к одному из стилей, основными из которых являются следующие: процедурный, функциональный, логический и объектно-ориентированный.

***Процедурное программирование***

Процедурное (императивное) программирование является отражением архитектуры традиционных ЭВМ, которая была предложена фон Нейманом в 40-х годах. Теоретической моделью процедурного программирования служит алгоритмическая система под названием «машина Тьюринга».

Программа на процедурном языке программирования состоит из последовательности операторов (инструкций), задающих процедуру решения задачи. Основным является оператор присваивания, служащий для изменения содержимого областей памяти. Концепция памяти как хранилища значений, содержимое которого может обновляться операторами программы, является фундаментальной в императивном программировании.

Выполнение программы сводится к последовательному выполнению операторов с целью преобразования исходного состояния памяти, то есть значений исходных данных, в заключительное, то есть в результаты. Таким образом, с точки зрения программиста имеются программа и память, причем первая последовательно обновляет содержимое последней.

Процедурные языки характеризуются следующими особенностями:

* необходимостью явного управления памятью, в частности, описанием переменных;
* малой пригодностью для символьных вычислений;
* отсутствием строгой математической основы;
* высокой эффективностью реализации на традиционных ЭВМ.

Одним из важнейших классификационных признаков процедурного языка является его уровень. Уровень языка программирования определяется семантической (смысловой) емкостью его конструкций и степенью его ориентации на программиста. Язык программирования частично ликвидирует разрыв между методами решения различного рода задач человеком и вычислительной машиной. Чем более язык ориентирован на человека, тем выше его уровень. Дадим краткую характеристику реализованным на ПЭВМ языкам программирования в порядке возрастания их уровня.

*Двоичный язык* является непосредственно машинным языком. В настоящее время такие языки программистами практически не применяются.

*Язык Ассемблера* — это язык, предназначенный для представления в удобочитаемой символической форме программ, записанных на машинном языке. Он позволяет программисту пользоваться мнемоническими кодами операций, присваивать удобные имена ячейкам и областям памяти, а также задавать наиболее удобные схемы адресации.

*Язык Макроассемблера* является расширением языка Ассемблера путем включения в него макросредств. С их помощью в программе можно описывать последовательности инструкций с параметрами — макроопределения. После этого программист может использовать снабженные аргументами макрокоманды, которые в процессе ассемблирования программы автоматически замещаются макрорасширениями. Макрорасширение представляет собой макроопределение с подставленными вместо параметров аргументами.

Другими словами, язык Макроассемблера представляет средства определения и использования новых, более мощных команд как последовательности базовых инструкций, что несколько повышает его уровень.

Языки Ассемблера и Макроассемблера применяются системными программистами-профессионалами с целью использования всех возможностей оборудования ЭВМ и получения эффективной по времени выполнения и по требуемому объему памяти программы. На этих языках обычно разрабатываются относительно небольшие программы, входящие в состав системного программного обеспечения: драйверы, утилиты и другие.

*Язык программирования С* (Си) первоначально был разработан для реализации операционной системы UNIX в начале 70-х годов. В последующем приобрел высокую популярность среди системных и прикладных программистов. В настоящее время этот язык реализован на большинстве ЭВМ.

В *С* сочетаются достоинства современных высокоуровневых языков в части управляющих конструкций и структур данных с возможностями доступа к аппаратным средствам ЭВМ на уровне, который обычно ассоциируется с языком низкого уровня типа языка Ассемблера. Язык С имеет синтаксис, обеспечивающий краткость программы, а компиляторы способны генерировать эффективный объектный код.

Одна из наиболее существенных особенностей С состоит в нивелировании различий между выражениями и операторами, что приближает его к функциональным языкам. В частности, выражение может обладать побочным эффектом присваивания, а также может использоваться в качестве оператора. Нет также четкой границы между процедурами и функциями, более того, понятие процедуры не вводится вообще.

Синтаксис языка затрудняет программирование и восприятие составленных программ. Отсутствует и строгая типизация данных, что предоставляет дополнительные возможности программисту, но не способствует написанию надежных программ.

*Ваsic* (Бэйсик) (Beginners All-purpose Symbolic Instruction Code) — многоцелевой язык символических инструкций для начинающих) представляет собой простой язык программирования, разработанный в 1964 году для использования новичками. Он был разработан как простейший язык для непосредственного общения человека с вычислительной машиной. Поэтому первоначально работа велась в интерактивном режиме с использованием интерпретаторов. В настоящее время для этого языка имеются также и компиляторы.

Согласно концепциям, заложенным в Basic, этот язык в смысле строгости и стройности является антиподом языка Pascal. В частности, в нем широко распространены различные правила умолчания, что считается плохим тоном в большинстве языков программирования подобного типа.

Basic широко распространен на ЭВМ различных типов и очень популярен в среде программистов, особенно начинающих. Существует множество диалектов этого языка, мало совместимых между собой. Basic активно поглощает многие концепции и новинки из других языков. Поэтому он достаточно динамичен, и нельзя однозначно определить его уровень.

*Pascal* (Паскаль) является одним из наиболее популярных среди прикладных программистов процедурным языком программирования, особенно для ПЭВМ. Разработанный в 1970 году швейцарским специалистом в области вычислительной техники профессором Н. Виртом, язык назван в честь французского математика и по замыслу автора предназначался для обучения программированию. Однако язык получился настолько удачным, что стал одним из основных инструментов прикладных и системных программистов при решении задач вычислительного и информационно-логического характера. В 1979 году был подготовлен проект описания языка — Британский стандарт языка программирования Pascal BS6192, который стал также и международным стандартом ISO 7185.

В языке Pascal реализован ряд концепций, рассматриваемых как основа «дисциплинированного» программирования и заимствованных впоследствии разработчиками многих языков. Одним из существенных признаков языка Pascal является последовательная и достаточно полная реализация концепции структурного программирования. Причем это осуществляется не только путем упорядочивания связей между фрагментами программы по управлению, но и за счет структуризации данных. Кроме того, в языке реализована концепция определения новых типов данных на основе уже имеющихся. Этот язык, в отличие от языка С, является строго типизированным. Pascal характеризуется:

* высоким уровнем;
* широкими возможностями;
* стройностью, простотой и краткостью;
* строгостью, способствующей написанию эффективных и надежных программ;
* высокой эффективностью реализации на ЭВМ.

Pascal реализован на ЭВМ различных типов, но наиболее распространен и развит для ПЭВМ. В настоящее время широко используются такие версии этого языка для ПЭВМ, как Borland Pascal и Turbo Pascal.

***Функциональное программирование***

Сущность функционального (аппликативного) программирования определена А. П. Ершовым как «... способ составления программ, в которых единственным действием является вызов функции, единственным способом расчленения программы на части является введение имени для функции, а единственным правилом композиции — оператор суперпозиции функции. Никаких ячеек памяти, ни операторов присваивания, ни циклов, ни, тем более, блок-схем, ни передачи управления».

Роль основной конструкции в функциональных языках играет выражение. К выражениям относятся скалярные константы, структурированные объекты, функции, тела функций и вызовы функций. Функция трактуется как однозначное отображение из Х в X, где Х — множество выражений.

Аппликативный язык программирования включает следующие элементы:

* классы констант, которыми могут манипулировать функции;
* набор базовых функций, которые программист может использовать без предварительного объявления и описания;
* правила построения новых функций из базовых;
* правила формирования выражений на основе вызовов функций.

Программа представляет собой совокупность описаний функций и выражения, которые необходимо вычислить. Данное выражение вычисляется посредством редукции, то есть серии упрощений, до тех пор, пока это возможно по следующим правилам: вызовы базовых функций заменяются соответствующими значениями; вызовы небазовых функций заменяются их телами, в которых параметры замещены аргументами.

Функциональное программирование не использует концепцию памяти как хранилища значений переменных. Операторы присваивания отсутствуют, вследствие чего переменные обозначают не области памяти, а объекты программы, что полностью соответствует понятию переменной в математике. В принципе, можно составлять программы и вообще без переменных. Кроме того, нет существенных различий между константами и функциями, то есть между программами и данными. В результате этого функция может быть значением вызова другой функции и может быть элементом структурированного объекта. Число аргументов при вызове функции не обязательно должно совпадать с числом параметров, указанных при ее описании. Перечисленные свойства характеризуют аппликативные языки как языки программирования очень высокого уровня.

Первым таким языком был LISP (LISTProcessing — обработка списков), созданный в 1959 году. Цель его создания состояла в организации удобства обработки символьной информации. Существенная черта этого языка — унификация программных структур и структур данных: все выражения записываются в виде списков.

***Логическое программирование***

Новую область — логическое, или реляционное программирование, — открыло появление языка PROLOG {Пролог} (PROgramminginLOGic — программирование в терминах логики). Этот язык был создан французским ученым А. Кольмероэ в 1973 году. В настоящее время известны и другие языки, однако наиболее развитым и распространенным языком логического программирования является именно Пролог. Так, имеется свыше 15 различных его реализации на ПЭВМ. Языки логического программирования, в особенности Пролог, широко используются в системах искусственного интеллекта, рассматриваемых в данном учебном пособии.

Центральным понятием в логическом программировании является отношение. Программа представляет собой совокупность определений отношений между объектами (в терминах условий или ограничений) и цели (запроса). Процесс выполнения программы трактуется как процесс общезначимости логической формулы, построенной из программы по правилам, установленным семантикой используемого языка. Результат вычисления является побочным продуктом этого процесса. В реляционном программировании нужно только специфицировать факты, на которых алгоритм основывается, а не определять последовательность шагов, которые требуется выполнить. Это свидетельствует о декларативности языка логического программирования. Она метко выражена в формуле Р. Ковальского: «алгоритм = логика + управление».

Языки логического программирования характеризуются:

* высоким уровнем;
* строгой ориентацией на символьные вычисления;
* возможностью инверсных вычислений, то есть переменные в процедурах не делятся на входные и выходные;
* возможной логической неполнотой, поскольку зачастую невозможно выразить в программе определенные логические соотношения, а также невозможно получить из программы все выводы правильные.

Логические программы, в принципе, имеют небольшое быстродействие, так как вычисления осуществляются методом проб и ошибок, поиском с возвратами к предыдущим шагам.

***Обьектно-ориентированое программирование***

Прототипом объектно-ориентированного программирования послужил ряд средств, входящих в состав языка SIMULA-67. Но в самостоятельный стиль оно оформилось с появлением языка SMALLTALK, разработанного А. Кеем в 1972 году и первоначально предназначенного для реализации функций машинной графики.

В основе объектно-ориентированного стиля программирования лежит понятие объекта, а суть его выражается формулой: «объект = данные + процедуры». Каждый объект интегрирует в себе некоторую структуру данных и доступные только ему процедуры обработки этих данных, называемые методами. Объединение данных и процедур в одном объекте называется инкапсуляцией и присуще объектно-ориентированному программированию.

Для описания объектов служат классы. Класс определяет свойства и методы объекта, принадлежащего этому классу. Соответственно, любой объект можно определить как экземпляр класса.

Программирование рассматриваемого стиля заключается в выборе имеющихся или создании новых объектов и организации взаимодействия между ними. При создании новых объектов свойства объектов могут добавляться или наследоваться от объектов-предков. В процессе работы с объектами допускается полиморфизм — возможность использования методов с одинаковыми именами для обработки данных разных типов.

К наиболее современным объектно-ориентированным языкам программирования относятся C++ и Java.

Язык C++ был разработан в начале 80-х годов Б. Страуструпом, сотрудником лаборатории Bell корпорации AT&T. Им была создана компактная компилирующая система, в которой за основу был взят язык С, дополненный элементами языков BCPL,Simula-67 и Algol-68. К июлю 1983 года появился язык С с классами, а чуть позднее — C++. К 1990 году была выпущена третья версия языка C++, принятая комитетом ANSI в качестве исходного материала для его стандартизации.

В 1990 году сотрудник корпорации Sun Д. Гослинг на основе расширения C++ разработал объектно-ориентированный язык Oak, основным достоинством которого было обеспечение сетевого взаимодействия различных по типу устройств. Новая интегрируемая в Internet версия языка, получила название Java. Первый броузер, который поддерживал язык Java, разработан программистом корпорации Sun П. Нафтоном и получил название HotJava. С января 1995 года Java получает распространение в Internet.

Согласно официальному определению авторов, Java является простым объектно-ориентированным и архитектурно-нейтральным языком интерпретирующего типа, обеспечивающим надежность, безопасность и переносимость, обладающим высокой производительностью в сочетании с многопоточностью и динамичностью.

Принципиальной разницей между Java и C++ является то, что первый из них является интерпретируемым, а второй — компилируемым. Синтаксис языков практически полностью совпадает.

С точки зрения возможностей собственно объектно-ориентированных средств язык Java обладает рядом преимуществ перед языком C++. Так, язык Java демонстрирует более гибкую и мощную систему инкапсуляции информации. Механизм наследования, реализованный в Java, обязывает к более строгому подходу к программированию, что улучшает надежность и понимаемость кода. Язык же C++ обладает сложной, неадекватной и трудной для понимания системой наследования. Возможности динамического связывания объектов одинаково хорошо представлены в обоих языках, однако, синтаксическая избыточность C++ заставляет и здесь отдать предпочтение языку Java.

В силу своей конструктивности идеи объектно-ориентированного программирования используются во многих универсальных процедурных языках. Так, например, в состав интегрированной системы программирования на языке PASCAL (корпорации Borland International) версии 5.5 входит специальная библиотека объектно-ориентированного программирования TurboVision.

В последнее время многие программы, в особенности объектно-ориентированные, реализуются как системы визуального программирования. Отличительной особенностью таких систем является мощная среда разработки программ из готовых «строительных блоков», позволяющая создать интерфейсную часть программного продукта в диалоговом режиме, практически без кодирования программных операций. К числу объектно-ориентированных систем визуального программирования относятся: VisualBasic, Delphi, C++ Builder и Visual C++.