1.8. Технологии программирования

Чтобы программы получались достаточно хорошими необходимо владеть некоторыми правилами их составления.

Существуют различные технологии проектирования программ:

Структурное программирование;

Логическое программирование;

Объектно-ориентированное;

Визуальное.

1.8.1. Структурное программирование

Все классические языки программирования являются языками процедурного типа. Это означает, что программист явно указывает все действия, которые должна выполнить ЭВМ.

С усложнением алгоритмов увеличиваются программы. Увеличение размера программ приводит к тому, что трудно определить, всегда ли программный продукт делает то, что нужно, и что он не делает ничего такого, что не требуется.

Одним из методов, улучшающих понимание программ, является *структурное программирование.* Этот метод позволяет основные усилия при разработке программ направить на наиболее подверженный ошибкам участок – логику программы.

Основные составляющие структурного программирования:

проектирование сверху вниз (нисходящая стратегия);

модульное программирование;

структурное кодирование.

Проектирование программного продукта сверху вниз предусматривает сначала определение задачи в общих чертах, а затем постепенное уточнение структуры путем внесения более мелких деталей. Проектирование представляет собой последовательность шагов такого уточнения. На каждом шаге необходимо выявить основные функции, которые нужно выполнить, т.е. задача разбивается на ряд подзадач, пока эти задачи не станут настолько простыми, что каждой из них будет соответствовать один программный модуль.

Иногда, чаще всего при отладке и тестировании программ, используют восходящую стратегию (снизу вверх). Сначала программируются и отлаживаются нижние в иерархии модули, а затем модули более высокого уровня.

Модульное программирование – это процесс разделения программы на логические части, называемые модулями, и последовательное программирование каждой части. Когда большая единая задача делится на подзадачи, то значительно проще разобраться в ее решении. Это обычный способ управления сложной ситуацией, основанный на принципе “разделяй и властвуй”.

При разбиении на модули необходимо, во-первых, добиваться того, чтобы программный модуль не зависел от контекста, в котором он будет использоваться, и, во-вторых, чтобы его можно было включать в программу без предварительных знаний о его внутреннем устройстве. Надо стремиться к тому, что бы ваши модули можно было использовать так же просто, как любую стандартную подпрограмму языка.

Каждый модуль должен иметь свое назначение, отличающееся от назначения других модулей. Разработка функционально независимых и минимально связанных между собой модулей позволяет избежать многих ошибок при программировании. Следует стремиться к независимости модулей так, чтобы изменение одного модуля не влекло за собой перепрограммирования других.

Структурное кодирование – это метод написания хорошо структурированных программ, который позволяет получать программы, более удобные для тестирования, модификации и использования. Он предполагает использование лишь разрешенных базовых структур, ограничение на вложенность условных операторов, на использование языковых конструкций с неочевидной семантикой, на величину модулей. Структурное программирование предполагает наличие только одного входа в модуль в начале и одного выхода в конце, документированность программ, соблюдение отступов и т. д. Хорошо структурированную программу можно читать сверху вниз без управляющих переходов на другие страницы.

Перечислим *цели* структурного программирования**:**

обеспечение дисциплины программирования – это основная цель структурного программирования и главный фактор достижения остальных его целей;

улучшение читабельности программ;

повышение эффективности программ – достигается за счет такого разбиения на модули, при котором легко находить и исправлять ошибки и переделывать текст любого модуля независимо от других;

повышение надежности программ – этого можно достичь, если программа легко поддается тестированию и отладке; а это возможно, если соблюдаются правила написания читабельных программ и хорошего структурирования при разбивке на модули;

уменьшение времени и стоимости разработки программ достигается при повышении производительности труда программистов; соблюдение же правил структурного программирования позволяет этого добиться.

Исходя из вышесказанного, перечислим ***правила***, которых нужно придерживаться при разработке программ:

программу нужно разбивать на модули;

необходимо аккуратно разрабатывать иерархическую структуру взаимосвязи между модулями;

ширина модуля (количество вызываемых им других модулей) не должна превышать десяти (это связано с тем, что в кратковременной памяти человека одновременно может храниться 5-9, в среднем 7, “кусков” информации), при превышении этого предела увеличивается количество ошибок;

модуль не должен содержать более 100 операторов;

модуль должен иметь одну входную и одну выходную точку;

каждый модуль должен быть документирован, т.е. содержать описание его назначения, входных и выходных данных, имена модулей, которые его вызывают и которые он вызывает, краткое описание алгоритма или ссылки на его описание, комментарии по ходу решения задачи;

следует избегать ненужных меток и переходов по оператору **goto**;

следует использовать только базовые структуры;

каждый оператор следует записывать в отдельной строке;

при записи составных операторов нужно использовать отступы;

нежелательна вложенность оператора **if** более 3;

следует избегать использования языковых конструкций с неочевидной семантикой и программистских “трюков”.

Мы рассмотрели основные базовые структуры, разрешенные в структурном программировании. Из них, как из кубиков, можно составить любой алгоритм.

На практике для проверки структурности блок-схемы можно использовать следующее правило.

Если, заменив последовательно каждую базовую структуру блок-схемы прямоугольником, в итоге получим один прямоугольник, то блок-схема является структурной.

Однако структурность блок-схемы не гарантирует правильность алгоритма: ведь и неправильный алгоритм можно изобразить структурно; с другой стороны, можно построить неструктурный алгоритм, дающий правильное решение поставленной задачи.

На рис.2.1 приведена абстрактная блок-схема, не удовлетворяющая принципам структурного программирования.

Иногда на практике оказывается полезным отступить от канонов структурного программирования. Но это допустимо в достаточно редко встречающихся случаях, к тому же у программистов нет единого мнения по этому поводу. Поэтому не будем начинать дискуссию на лекции, а перенесем обсуждение данной темы на практические занятия.

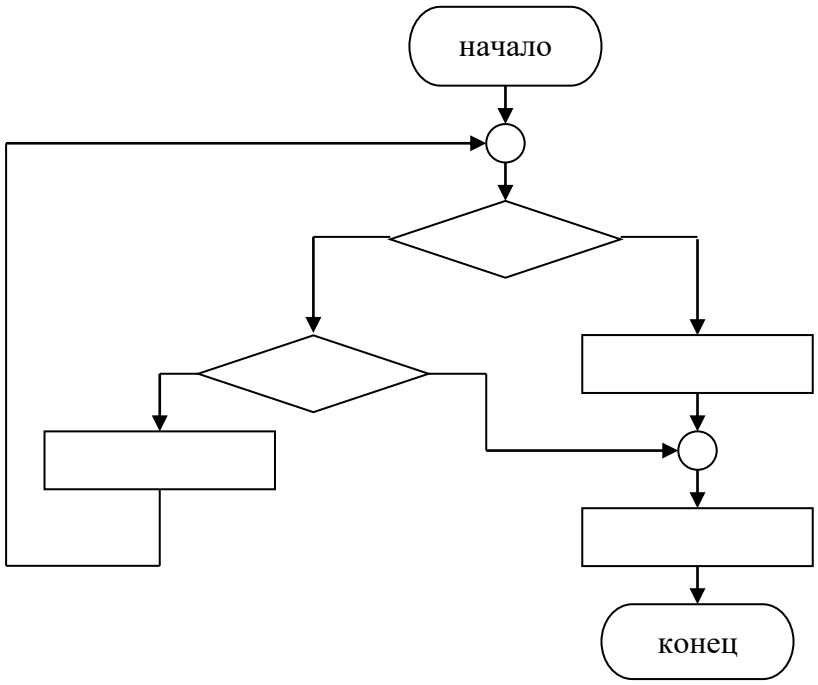

Рис. 2.1. Пример неструктурной блок-схемы

1.8.2 Логическое программирование

Суть его в том, что компьютеру предоставляется не алгоритм решения задачи, а формальное описание предметной области и решаемой задачи. Делается это с помощью логических формул. Вывод решения осуществляется компьютером. Примером реализации логического программирования является система Пролог. В научных исследованиях используется язык Лисп.

Логическое программирование в основном используется для решения задач в области искусственного интеллекта (обработка текстов на ЕЯ, построение ЭС, создание музыкальных произведений, понимание речи и т.д.).

1.8.3. Объектно-ориентированное программирование

Это новый этап в развитии современных концепций построения языков программирования (в 80-х годах).

В нем получили дальнейшее развитие принципы структурного программирования (структуризация программ и данных, модульность, …), а также новые подходы к созданию программ.

ООП основано на понятии объекта – типа данных, в котором сочетаются свойства объекта и методы его обработки.

Примеры:

Объект автомобиль, свойства его – марка, тип двигателя, методы – способность двигаться в соответствии со значением своих свойств (количество бензина, мощность двигателя…).

Объект файл, свойства – имя, размер, методы – читать, писать, создавать, удалять.

ООП есть в Паскале, начиная с версии 5.0, в C++. Подробнее рассмотрим позже.

1.8.4. Визуальное программирование

Визуализация – это процесс отображения на экране компьютера сложных процессов построения программ.

Визуальное программирование как бы добавляет новое измерение при создании приложений, давая возможность изображать эти объекты на экране до выполнения самой программы.

Без визуального программирования процесс отображения требует написания фрагмента кода, создающего и настраивающего объект «по месту». Увидеть закодированные объекты было возможно только в ходе исполнения программы

Частично визуальное программирование реализовано в Delphi, Visual C++. В этих системах визуализирована работа с элементами интерфейса.

1.9. Способы описания языков программирования

Грамматическое описание любого языка включает в себя его алфавит, синтаксис и семантику.

Алфавит – набор основных символов (букв).

Синтаксис – правила построения слов и фраз (предложений).

Семантика – определяет смысловое значение фраз.

Нормальная форма Бэкуса-Наура (НФБ)

В НФБ каждая формула представляет собой определение некоторого термина языка. Слева в формуле записывается в угловых скобках определяемый термин, затем знак "::=", который соответствует слову "есть", а в правой части формулы определяется значение термина. Слова, записанные без угловых скобок обозначают "термы", т.е. конечные понятия, не подлежащие определению. Знак "|" – вертикальная черта обозначает "или".

Например:

```
<цифра> ::= 0|1|2|3|4|5|6|7|8|9
<буква> ::= A|B|C|D|E|F|G|H|…|Z|a|b|c|d|e|f|g|h|…|z
<символ_подчеркивания> ::= _
<идентификатор> ::= <буква> | <символ_подчеркивания> |
 <идентификатор><буква> | <идентификатор><цифра> |
 <идентификатор><символ_подчеркивания >
```

Такие формулы формального описания языка называются *металингвистическими*.

В примере 3 использовано *рекурсивное* определение (сам определяемый термин используется и в правой части формулы).

Синтаксические диаграммы

Смотрите конспект лекций!!!