Лекция 1

Тема 1

Предмет и содержание курса.

Алгоритмизация задач и программирование

Учебная дисциплина

Информатика: Алгоритмы и языки программирования

Краткая справка дисциплины

Информатика: Алгоритмы и языки программирования

Б1.Б.23.

Направление подготовки: 03.03.03 - Радиофизика

Форма промежуточного контроля дисциплины - зачет

Преподаватель:

доцент кафедры радиоастрономии
кандидат физико-математических наук

Колчев Алексей Анатольевич

e-mail: kolchevaa@mail.ru
ауд. 1303 (каф. радиоастрономии)

Литература

1. В.В.Подбельский, С.С.Фомин. Программирование на языке Си. М.: Финансы и статистика, 2003.- 600 с.
2. Павловская Т.А. С/С++. Программирование на языке высокого уровня. СПб.: Питер, 2002, 2005, 2007
3. Колдаев В.Д. Численные методы и программирование: Учебное пособие / В.Д. Колдаев; Под ред. Л.Г. Гагариной. - М.: ИД ФОРУМ: НИЦ ИНФРА-М, 2014. - 336 с.: ил.; 60x90 1/16. - (Профессиональное образование). (переплет) ISBN 978-5-8199-0333-9, 300 экз. Режим доступа: - http://znanium.com/bookread.php?book=452274
4. Язык Си++ : учебное пособие для студентов высших учебных заведений, обучающихся по направлениям "Прикладная математика" и "Вычислительные машины, комплексы, системы и сети" / В. В. Подбельский .? 5-е изд. ? Москва : Финансы и статистика, 2008 .? 559 с. : ил. ; 21 .? Библиогр.: с. 538-539 (35 назв.) .? Указ. символов, предм. указ.: с. 540-556 .? ISBN 978-5-279-02204-5, 3000. 256
5. Бахвалов, Н. С. Численные методы [Электронный ресурс] / Н. С. Бахвалов, Н. П.Жидков, Г. М. Кобельков. - 7-е изд. (эл.). - М. : БИНОМ. Лаборатория знаний, 2012. - 636 с. : ил. - (Классический университетский учебник). - ISBN 978-5-9963-0802-6. Режим доступа: - http://e.lanbook.com/view/book/4397/page83/

Определение

- **Информатика — научная дисциплина, изучающая процессы обработки информации с использованием ЭВМ, то есть процессы получения, передачи, представления, хранения и обработки информации, а также технические и программные средства ЭВМ.**

Логические основы ЭВМ, элементы и узлы

- **Основа: алгебра логики**

Алгебра логики – это раздел математической логики, значения всех элементов (функций и аргументов) которой определены в двухэлементном множестве: 0 и 1. Алгебра логики оперирует логическими высказываниями.

- **Основные технические элементы: триггеры, сумматоры, регистры, шифраторы, дешифраторы, мультиплексоры, демультиплексоры, счетчики.**

Комбинационные и последовательные схемы

Преобразование информации в ЭВМ производится электронными устройствами двух классов: комбинационные и последовательные схемы (цифровые автоматы). В комбинационной схеме совокупность выходных сигналов в любой момент времени однозначно определяется входными сигналами. Такой способ обработки называется комбинационным, так как результат зависит только от комбинации входных сигналов и вырабатывается сразу при подаче входной информации. Закон функциональности комбинационной схемы определён, если задано соответствие между её входными и выходными словами.

Комбинационные и последовательные схемы

Цифровой автомат, в отличие от комбинационной схемы имеет конечное число различных состояний. Под воздействием входного слова цифровой автомат переходит из одного состояния в другое и выдаёт выходное слово. Оно определяется входным словом, поступившим в этот такт на вход автомата и внутренним состоянием автомата, которое явилось результатом воздействия на автомат входного слова предыдущего этапа.

Цифровой автомат содержит память, состоящую из запоминающих элементов, триггеров, элементов задержки и других элементов, фиксирующих состояние, в котором он находится.

Комбинационная схема не содержит запоминающих элементов, поэтому её называют автоматом без памяти или примитивным автоматом.

На рисунке 1., изображен цифровой сигнал.

Рисунок 1.

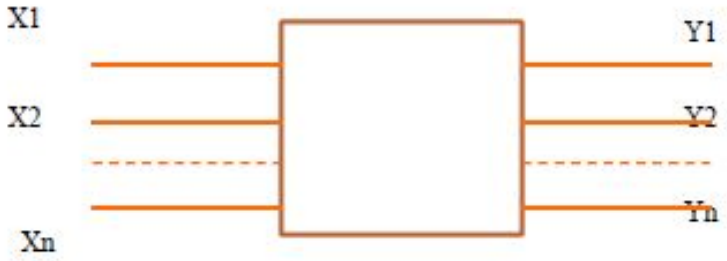


Рисунок 2.

На рисунке 2., изображено условное обозначение простейшего цифрового автомата. На входы X1, X2, …Xn поступают цифровые сигналы и преобразуются в соответствии с функцией логического преобразования в сигналы Y1,Y2,…Yn.

Основные элементы алгебры логики

Анализ комбинационных устройств и цифровых автоматов проще всего проводить с помощью алгебры логики, оперирующей только двумя понятиями: истинным (логическая 1) и ложным (логический 0).

Алгебру логики называют еще **булевой алгеброй**.

Простейшими операциями в алгебре логики являются операции логического сложения (иначе: операция **ИЛИ (OR),** операция дизъюнкции) и логического умножения (иначе: операция **И (AND)**, операция конъюнкции) и операция логического отрицания (иначе: **НЕ (NOT),** операция инверсия).

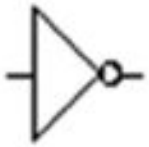
Основные элементы алгебры логики

Логическими элементами компьютеров являются электронные схемы :

Логические элементы И (AND) и И-НЕ (NAND)

Логические элементы ИЛИ (OR) и ИЛИ-НЕ (NOR)

Инвертирующий элемент НЕ (NOT)



Исключающее ИЛИ (XOR)

Основные элементы алгебры логики

Логический элемент Исключающее ИЛИ (XOR) реализует логическую операцию неравнозначность (или суммирование по модулю два). Эта функция может быть получена, используя простейшие операции алгебры логики. Логические элементы в схемах называют вентилями.

Рассмотрим логические операции и приведем их таблицы истинности.

Основные элементы алгебры логики

1) Конъюнкция (логический элемент И (AND)) ее обозначают & или × или ^

Высказывание A&B истинно в том случае, когда истинны оба входящих в него высказывания.

A	B	A&B
0	0	0
0	1	0
1	0	0
1	1	1

Основные элементы алгебры логики

2) Дизъюнкция (логический элемент ИЛИ (OR)) ее обозначают + или ∨

Высказывание A+B истинно, когда хотя бы истинно одно из входящих высказываний.

A	B	A+B
0	0	0
0	1	1
1	0	1
1	1	1

Основные элементы алгебры логики

3) Инверсия (логический элемент НЕ (NOT) и обозначается ¯ (черточкой сверху) или ′ (запятая сверху).

Присоединение частицы НЕ к некоторому высказыванию называется отрицанием и если высказывание истинно, то с инверсией оно ложно и наоборот.

A	A’
0	1
1	0

Основные элементы алгебры логики

4) Исключающее ИЛИ (логический элемент XOR) обозначается $\oplus$.

Высказывание $A \oplus B$ истинно тогда когда одно высказывание истинно, а второе ложное.

A	B	$A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0

На рисунке 3 представлена схема логического выражения Y=A×B’×C+A×B×C (входы X1,X2, X3 обозначить через A,B,C)

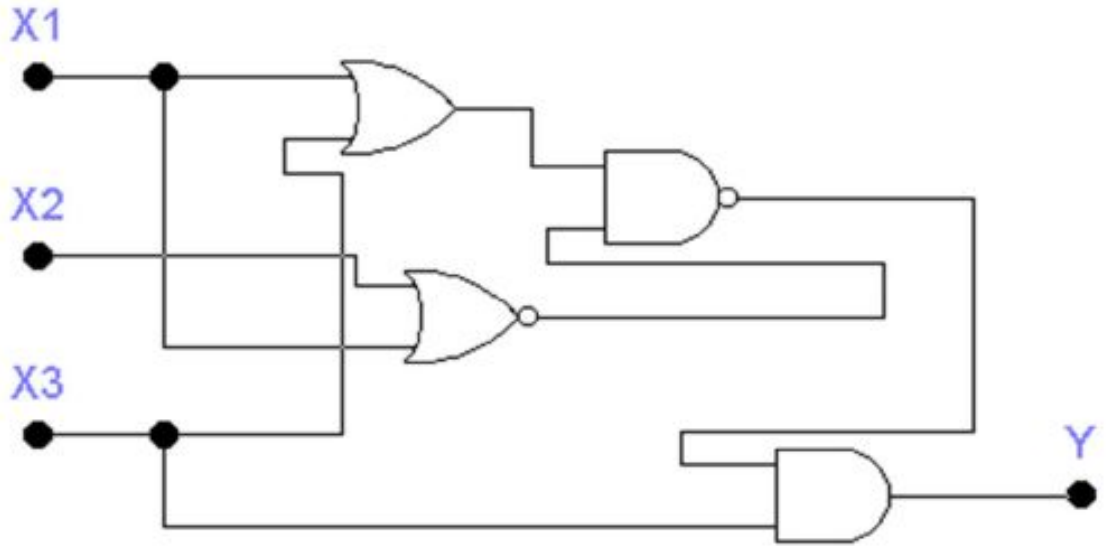


Рисунок 3.

С помощью схем логических элементов можно реализовать любую логическую функцию, описывающую работу устройств компьютера. Обычно у вентилей от 2 до 8 входов и один выход.

Триггеры

Триггер - это устройство последовательного типа с двумя устойчивыми состояниями равновесия, предназначенное для записи и хранения информации. Под действием входных сигналов триггер может переключаться из одного устойчивого состояния в другое.

Триггер — элементарная ячейка оперативной памяти.

В асинхронных триггерах информация может записываться непрерывно и определяется информационными сигналами, действующими на входах в данный момент времени. Если информация заносится в триггер только в момент действия так называемого синхронизирующего сигнала, то такой триггер называют

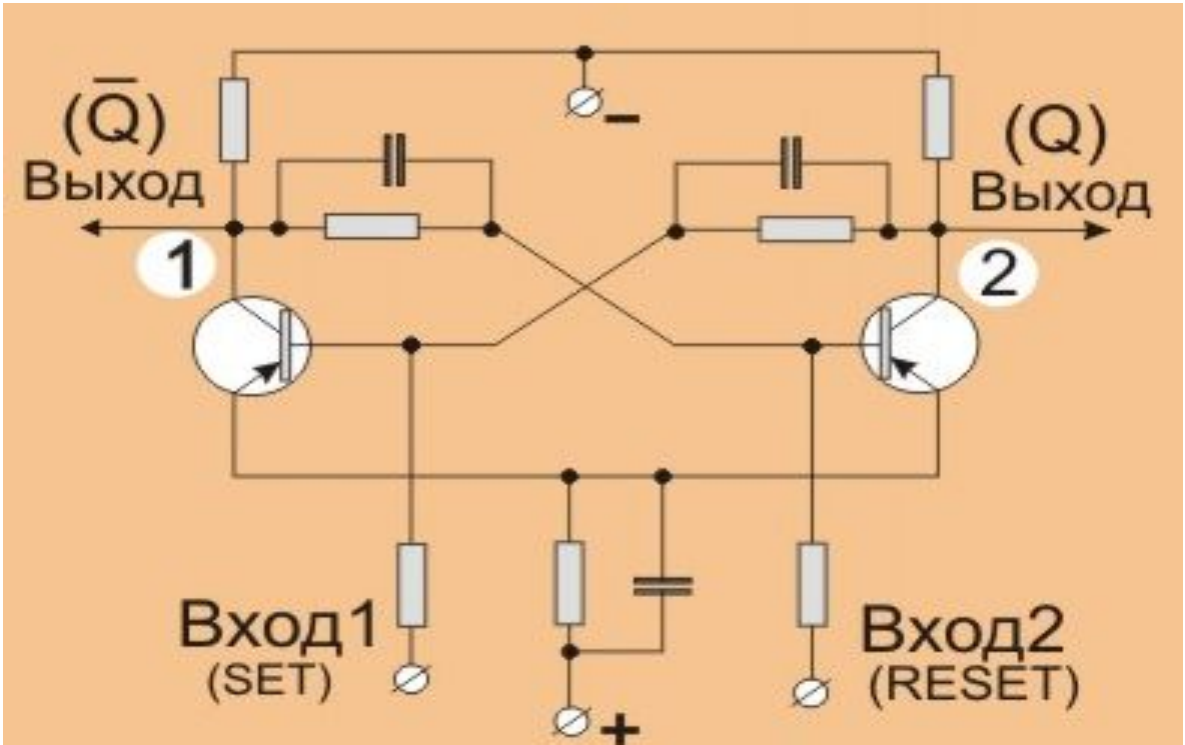
Триггеры

Несколько триггеров можно объединить в **регистр** - узел для хранения чисел с двоичным представлением цифр разрядов.

В цифровой технике приняты следующие обозначения входов триггеров:
S(set) - раздельный вход установки в единичное состояние (напряжение высокого уровня на прямом выходе Q);
R(reset) - раздельный вход установки в нулевое состояние (напряжение низкого уровня на прямом выходе Q);
D - информационный вход (на него подается информация, предназначенная для занесения в триггер);

Триггеры

Простейшая электронная схема триггера состоит из двух усилительных каскадов. Выход каждого из каскадов подключен к входу другого через резисторы. Номиналы этих резисторов подобраны так, что каскад с полностью открытым транзистором, уверенно запирает транзистор другого каскада. Если подать на триггер питающее напряжение, то оба каскада начинают "бороться" между собой, пытаясь закрыть друг-друга.



Триггеры

Как бы не были транзисторы близки по характеристикам, один из них(присвоим ему номер1) обязательно окажется "сильнее" и закроет другой (для удобства обозначим его как номер 2) Все происходит очень быстро, выглядит так, что транзистор 1 мгновенно оказывается открытым, а другой (2) закрытым. В таком состоянии триггер может находиться очень долго. Можно назвать его - 1-м устойчивым состоянием.

Если подать на вход закрытого каскада(2) имульс напряжения, достаточный, что бы его открыть на короткое время, то открывшись он "запрет" каскад 1, пребывающий до этого момента в открытом состоянии. Закрывшись, каскад 1 перестает запирать каскад 2, и тот так и останется открытым. Таким образом, каскады поменяются местами, триггер окажется во 2-м устойчивом состоянии.

Триггеры

В таком состоянии он может находиться очень долго, если не подать открывающий импульс, на закрытый каскад 1. Каскад 1 открываясь, запрет каскад 2 и триггер вернется в первоначальное состояние(1). Получается, что наш триггер имеет два устойчивых состояния и два управляющих входа, подав на которые импульсы достаточной амплитуды, можно эти состояния менять.

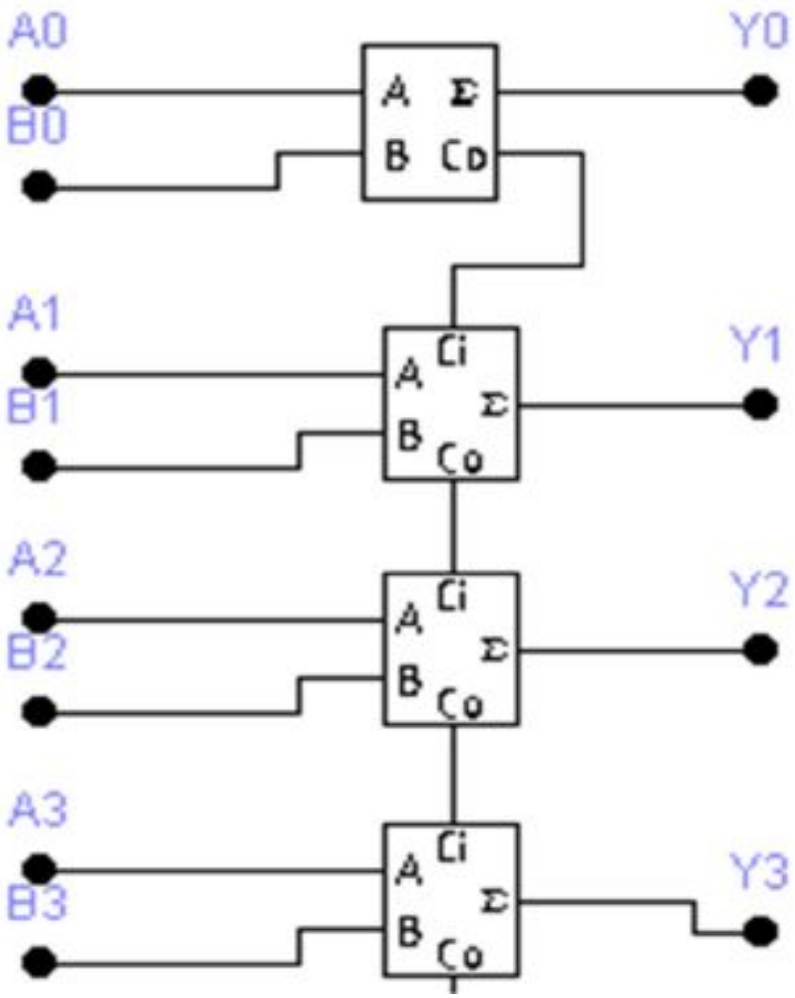
Сумматоры

Сумматор предназначен для арифметического сложения двух n-разрядных двоичных чисел. Простейшим элементом, реализующим операцию сложения двух одноразрядных двоичных чисел является элемент Исключающее ИЛИ. При этом сложение двух логических 1 дает 0 как цифру суммы и перенос в следующий разряд.

Полный сумматор дополнительно учитывает возможность переноса из младшего разряда.

Сумматоры

Многоразрядный сумматор создается на базе одного полусумматора (младший разряд) и полных сумматоров. На рисунке представлен четырехразрядный сумматор.
Назначение выводов: A, B - входы слагаемых, Σ- результат суммирования, Ci - выход переноса, Co - вход переноса.



Этапы решения задачи на ЭВМ

Обычно можно выделить несколько этапов, характерных для большинства задач, решаемых с помощью ЭВМ:

- постановка задачи;
- математическая формулировка задачи;
- построение алгоритма решения;
- запись алгоритма в виде программы;
- ввод программы в компьютер и трансляция;
- отладка программы;
- эксплуатация программы.

Рассмотрим отдельные этапы более подробно.

1 этап. На этом этапе вырабатывается точная формулировка цели задачи, определяется роль ЭВМ в ее реализации и целесообразность применения компьютера вообще. От корректности постановки задачи пользователем в значительной мере зависят все последующие этапы и успех решения задачи в целом.
2 этап. Осуществляется формализация описания задачи, т. е. с помощью математических, статистических и других методов соотношения между величинами выражаются с помощью математических, логических формул. Вообще, представление объекта исследования (задачи) в

Построение такой модели часто является предметом исследования целого ряда дисциплин: исследование операций, методы операций, математическая статистика, теория информации, численный анализ и др. Впервые метод математического моделирования реальных явлений возник и получил свое развитие в физике (первой серьезной математической моделью здесь можно считать классическую физику Ньютона). При этом следует принимать во внимание и определенные параметры компьютера как инструмента решения: скорость выполнения операций, точность вычисления, объем памяти и т. д.

Построение алгоритма решения

3 этап. Остановимся вначале на некоторых определениях понятия «**алгоритм**».

1. Конечная последовательность действий, исполнение которых позволяет за конечное время получить решение некоторой задачи или любой задачи из некоторого класса задач.
2. Понятное и точное предписание исполнителю совершить определенную последовательность действий для достижения поставленной цели.
3. Точное предписание, определяющее вычислительный процесс, преобразующий исходные данные в искомый результат.

Таким образом алгоритмом называется последовательность действий определяющих некоторый процесс ведущий от некоторых начальных данных к искомому результату.

Данные - факты и идеи, представленные в формализованном виде, позволяющем передавать и обрабатывать эти факты и идеи с помощью некоторого процесса.

Алгоритму присущи определенные свойства.

- *определенность алгоритма* - однозначность выполнения каждого отдельного шага преобразования информации;
- *выполнимость* - конечность действий алгоритма решения задач, позволяющая получить желаемый результат при допустимых исходных данных за конечное число шагов;
- *массовость* - пригодность алгоритма для решения определенного класса задач.

В алгоритме обязательно должны быть предусмотрены все ситуации, которые могут возникнуть в процессе решения комплекса задач.

На практике наиболее часто используются три способа записи алгоритма: словесный; графический; программный.

Отдельные этапы алгоритма обычно нумеруются.

1-й способ: естественная (словесная) запись алгоритма.

Примеры. Алгоритм попадания в свою квартиру

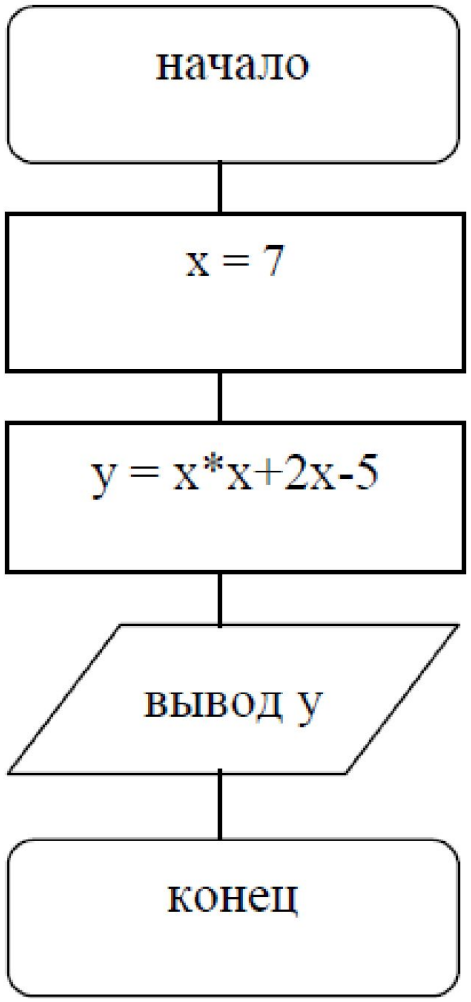
1) подойти к двери своей квартиры;
2) достать ключ;
3) вставить в замочную скважину;
4) повернуть n раз;
5) вынуть ключ;
6) распахнуть дверь
7) зайти в квартиру.

Алгоритм вычисления выражения $y = x*x+2x-5$ для $x = 7$

1) присвоить переменной x значение 7;
2) подставить значение x в приведенное выражение и произвести вычисления;
3) потребовать ответ;
4) прекратить вычисления.

2-й способ: запись алгоритма в графической форме. Обычно под этим понимается изображение алгоритма в виде блок-схемы.

Пример. Так выглядит блок-схема алгоритма математической задачи, приведенного в предыдущем примере



3-й способ: запись алгоритма на конкретном языке программирования. Организация такой записи оставляет основной смысл следующего, четвертого этапа решения задач на ЭВМ.

Методы разработки и анализа алгоритмов
Нисходящим проектированием алгоритмов называется такой метод составления алгоритмов, когда исходная задача разбивается на ряд вспомогательных подзадач или подалгоритмов, решаемых с использованием более простых и элементарных операций или процедур. Последние, в свою очередь, вновь разбиваются на более простые и элементарные, и так до тех пор, пока не дойдём до команд исполнителя.
Восходящий метод, наоборот, опираясь на некоторый, заранее определяемый корректный набор подалгоритмов, строит функционально завершенные подзадачи более общего назначения, от них переходит к более общим, и так далее, до тех пор, пока не дойдем до уровня, на котором можно записать решение поставленной задачи. Этот метод известен как метод проектирования "снизу вверх".

Структурные принципы алгоритмизации – это принципы формирования алгоритмов из базовых структурных алгоритмических единиц (следование, ветвление, повторение), используя их последовательное соединение или вложение друг в друга с соблюдением определённых правил, гарантирующих читабельность и исполняемость алгоритма сверху вниз и последовательно.
Структурированный алгоритм – это алгоритм, представленный как следования и вложения базовых алгоритмических структур. У структурированного алгоритма статическое состояние (до реализации алгоритма) и динамическое состояние (после реализации) имеют одинаковую логическую структуру, которая прослеживается сверху вниз ("как читается, так и исполняется"). При структурированной разработке алгоритмов правильность алгоритма можно проследить на каждом этапе его построения и

Теорема о структурировании: Любой алгоритм может быть эквивалентно представлен структурированным алгоритмом, состоящим из базовых алгоритмических структур.
Последовательное выполнение операторов в языке программирования означает, что они выполняются в том порядке, как записаны в программе.
Условное выполнение означает, что при определенных условиях должны выполняться одни операторы, а при других условиях - другие. Для организации такого порядка служат условные операторы. Условное выполнение также часто называют ветвлением. При повторяющемся выполнении некоторые операторы выполняются несколько раз, несмотря на то, что в программе они записаны один раз) обеспечивается операторами цикла).
Одним из широко используемых методов проектирования и разработки алгоритмов (программ) является модульный метод (модульная

Модуль – это некоторый алгоритм или некоторый его блок, имеющий конкретное наименование, по которому его можно выделить и актуализировать. Иногда модуль называется вспомогательным алгоритмом, хотя все алгоритмы носят вспомогательный характер. Это название имеет смысл, когда рассматривается динамическое состояние алгоритма; в этом случае можно назвать вспомогательным любой алгоритм, используемый данным в качестве блока (составной части) тела этого динамического алгоритма. Используют и другое название модуля – подалгоритм. В программировании используются синонимы – процедура, подпрограмма.

Свойства модулей:

- функциональная целостность и завершенность (каждый модуль реализует одну функцию, но реализует хорошо и полностью);
- автономность и независимость от других модулей (независимость работы модуля-преемника от работы модуля-предшественника; при этом их связь осуществляется только на уровне передачи/приема параметров и управления);
- эволюционируемость (развиваемость);
- открытость для пользователей и разработчиков (для модернизации и использования);
- корректность и надежность;

Ссылка на тело модуля происходит только по имени модуля, то есть вызов и актуализация модуля возможны только через его заголовок.

Преимущества модульного проектирования алгоритмов:

- возможность разработки алгоритма большого объема (алгоритмического комплекса) различными исполнителями;
- возможность создания и ведения библиотеки наиболее часто используемых алгоритмов (подалгоритмов);
- облегчение тестирования алгоритмов и обоснования их правильности ;
- упрощение проектирования и модификации алгоритмов ;
- уменьшение сложности разработки (проектирования) алгоритмов (или комплексов алгоритмов);
- наблюдаемость вычислительного процесса при реализации алгоритмов.

4 этап. Запись алгоритма в виде программы

Очевидно, что алгоритм решения задачи, предназначенный для исполнения на ЭВМ, должен быть записан на понятном компьютеру языке, языке программирования.

- **Язык программирования** - формализованный язык (набор символов и система правил образования и истолкования конструкций из этих символов), предназначенный для описания алгоритмов решения задач на ЭВМ.

Деятельность, включающая в себя запись алгоритма решения на конкретном языке программирования, а также выбор структуры используемых в ходе решения данных, называют программированием. Результатом такой

- Программа - упорядоченная последовательность предложений языка программирования (инструкций), описывающих алгоритм решения задачи.
- Очевидно, что каждое предложение программы должно заставлять компьютер выполнять определенную последовательность действий. При решении задач на ЭВМ 1-го поколения (40-е и начало 50-х годов XX века) программы записывались на т. н. машинном языке. Это означает, что каждая инструкция программы записывалась на языке внутреннего кодирования информации, т. е. чаще всего представляла последовательность нулей и единиц.

Такое программирование:

- а) очень трудоемко;
- б) не наглядно (трудно понять по тексту программы, что она, собственно, делает);
- в) не эффективно (для выполнения одного и того же алгоритма на ЭВМ с различной архитектурой приходится создавать разные программы).

Поэтому в середине 50-х годов появились первые языки программирования, использующие символику, близкую общепринятой математической.

- **Язык программирования** – знаковая система, предназначенная для описания элементов алгоритма. Язык программирования определяет набор правил, используемых при составлении компьютерной программы.
- Если язык программирования ориентирован на конкретный тип процессора и учитывает его особенности, то он называется языком программирования низкого уровня. К таким языкам относят язык Ассемблер, который представляет каждую команду машинного кода с помощью символьных условных обозначений и конкретной компьютерной архитектуре соответствует свой язык ассемблера.

- С развитием программирования появились языки, ориентированные на более высокий уровень абстракции при описании решаемой на ЭВМ задачи. Эти языки получили название языков высокого уровня. Их теоретическую основу составляют алгоритмические языки, например: Паскаль, Си, Фортран. Языки программирования высокого уровня значительно ближе и понятнее человеку, чем компьютеру (особенности конкретных компьютерных архитектур в них не учитываются).
- Для перевода программы, написанной на языке высокого уровня, в соответствующую машинную программу используются языковые процессоры.

- Различают два вида языковых процессоров: интерпретаторы и трансляторы. **Интерпретатор** – это программа, которая получает исходную программу и по мере распознавания конструкций входного языка реализует действия, описываемые этими конструкциями. **Транслятор** – это программа, которая принимает исходную программу и порождает на выходе программу, записанную на машинном языке. Транслятор с языка высокого уровня называют **компилятором**.

- *Откомпилированные* программы работают быстрее, но *интерпретируемые* проще исправлять и изменять. Каждый конкретный язык ориентирован либо на компиляцию, либо на интерпретацию — в зависимости от того, для каких целей он создавался. Например, *СИ* обычно используется для решения довольно сложных задач, в которых важна скорость работы программ. Поэтому данный язык обычно реализуется с помощью *компилятора*.
- Иногда для одного языка имеется *и компилятор, и интерпретатор*. В этом случае для разработки и тестирования программы можно воспользоваться интерпретатором, а затем откомпилировать отлаженную программу, чтобы

Пример: листинг программы вычисления выражения y = x*x + 2x -5 для значения x = 7, составленной на языке Си:

```
#include <stdio.h>                          /*начало главной функции*/
int main()                                  /*начало главной функции*/
{
float x,y;                                  /* объявление переменных */
x=7.;                                       /*задание значения x*/
y= x*x+2*x-5;                               /*вычисление У*/
printf("x=%3.1f y=%8.5f \n",x,y); /*вывод переменных*/
return 0;                                   /*конец главной функции*/
}
```

5 этап. Основное назначение этого этапа - ввод программы в компьютер. Если текст программы написан на машинонезависимом языке, то для ее выполнения требуется перевести машинный на язык. Этот перевод осуществляется автоматически, если в состав программного обеспечения компьютера входит специальная программа - транслятор с данного языка.
6 этап. С устранения из программы с помощью транслятора всех синтаксических ошибок начинается один из наиболее важных этапов работы с программой - ее отладка.
Отладка - процесс поиска, обнаружения (локализации) и устранения ошибок в программе. В общем случае, все ошибки можно разделить на три группы:

- Синтаксические ошибки
- Ошибки выполнения
- Логические ошибки

- На этапе отладки следует предусмотреть тщательное тестирование программы. Тест содержит набор исходных данных, для которых решение задачи известно. Если в ходе выполнения теста получаются результаты, отличные от ожидаемых, это свидетельствует о наличии логических ошибок в программе. Тесты также позволяют установить границы применимости тестируемой программы.