Оглавление

Введение 8
1. АЛГОРИТМИЗАЦИЯ И ТЕХНОЛОГИИ ПРОГРАММИРОВАНИЯ 10
1.1. Определение алгоритма и его свойства 10
1.2. Способы записи и средства описания алгоритма 11
1.3. Рисование блок-схемы алгоритма в приложениях Microsoft Office 14
1.4. Этапы решения прикладных задач на компьютере 16
1.5. Жизненный цикл программного продукта 18
1.6. Технологии и языки программирования 20
1.7. Трансляция, компиляция и интерпретация программ 26
1.8. Основные понятия языков программирования 27
1.8.1. Линейная структура 29
1.8.2. Разветвляющаяся структура 30
1.8.3. Циклическая структура 32
2. ОСНОВЫ ПРОГРАММИРОВАНИЯ НА ЯЗЫКЕ PASCAL 37
2.1. Представление информации в различных системах счисления 37
2.2. Основы двоичного кодирования в машинной арифметике 39
2.3. Организация программ линейной структуры. Ввод и вывод данных 42
2.4. Организация программ разветвляющейся структуры 46
2.5. Организация программ циклической структуры 47
2.5.1. Алгоритмы обработки одномерных массивов 49
2.5.2. Алгоритмы обработки двумерных массивов 51
2.6. Записи в Pascal 53
2.7. Алгоритмы обработки данных строкового типа 55
2.8. Управление экраном и звуком компьютера в Turbo Pascal 57
2.9. Графика в Pascal 61
2.10. Работа с формами в Pascal ABC 63
3. ОСНОВЫ ПРОГРАММИРОВАНИЯ В СРЕДЕ VBA 66
3.1. Основы объектно-ориентированного программирования 66
3.2. Интегрированная среда разработки VBA 69
3.3. Основные конструкции языка программирования VBA 76
3.3.1. Технологии ввода и вывода данных на VBA 80
3.3.2. Использование объектов для работы с диапазонами ячеек 83
4. ПРАКТИКУМ ПО ОСНОВНЫМ АЛГОРИТМИЧЕСКИМ КОНСТРУКЦИЯМ VBA 86
4.1. Программирование линейного вычислительного процесса 86
4.2. Создание и редактирование макросов с помощью макрорекордера 97
4.3. Работа с объектом Range: чтение и запись данных на рабочем листе 103
4.3.1. Переменные объектов 106
4.3.2. Использование оператора With 108
4.3.3. Методы и свойства объекта Range 109
4.4. Программирование разветвляющихся вычислительных процессов 112
4.4.1. Оператор условия If 112

4.4.2. Оператор выбора Select Case 117
4.5.Разработка программ для циклического вычислительного процесса 122
4.5.1. Вычисляемый цикл 122
4.5.2. Программирование табулирования функции 124
4.5.3. Циклы с условием 127
4.5.4. Смешанные циклы 132
4.5.5. Вложенные циклы 133
4.6.Обработка массивов на языке VBA 134
4.6.1. Цикл объектного типа For Each…Next для массивов 136
4.6.2. Автоматизация обработки массивов с Range 137
4.6.3. Алгоритмы обработки одномерных массивов 142
4.6.4. Программная обработки одномерных массивов на рабочем листе 143
4.6.5. Проекты обработки одномерных массивов в циклах с условием 148
4.6.6. Сравнение обработки массивов и данных пользовательского типа 157
4.6.7. Алгоритмы обработки двумерных массивов 160
4.6.8. Вывод массива в несколько колонок на окне MsgBox 165
4.7.Пользовательские процедуры и функции 165
4.8.Технологии вычислений в VBA 173
4.8.1. Функция Array для обработки массивов 173
4.8.2. Функция Split формирования массива из строковой переменной 175
4.8.3. Запись двумерного массива с использованием вложенных литералов 176
4.8.4. Использование функций рабочего листа Excel в VBA 178
5. РАЗРАБОТКА ГРАФИЧЕСКОГО ИНТЕРФЕЙСА ПРОЕКТА В VBA 181
5.1.Использование элементов управления на рабочем листе MS Excel 181
5.1.1. Расчеты в Excel с помощью кнопки экранного интерфейса
и клавишного макроса 182
5.1.2. Проект расчета стоимости заказа с элементами управления формы 188
5.1.3. Пример ввода/вывода данных с элементами управления формы 193
5.2.Работа с элементами управления на форме пользователя UserForm 196
5.3.Примеры проектов с управляющими элементами Label, TextBox, Image .. 201
5.3.1. Разработка проекта контроля веса 203
5.3.2. Проект выбора наименьшего числа из одномерного массива 204
5.3.3. Имена констант в обмене данными между формой и таблицей 205
5.3.4. Вычисление площади треугольника по формуле Герона 206
5.3.5. Численные методы решения нелинейных уравнений 208
5.3.6. Метод золотого сечения – численный метод поиска экстремума 218
5.3.7. Схема Горнера для решения полиномиальных уравнений 224
5.3.8. Численные методы вычисления интегралов 228
5.4.Примеры применения элемента ListBox 232
5.4.1. Автоматизация обработки массивов данных в VBA 234
5.4.2. Проект создания корзины и расчета стоимости покупки 239
5.4.3. Создание списка ListBox из нескольких колонок 242
5.4.4. Нахождение суммы членов ряда в Microsoft Excel 243
5.5.Технологии оформления главной страницы приложения (проекта) 251

5.5.1. Пример применения элементов Frame, OptionButton, CheckBox и ScrollBar для расчета стоимости покупки 251
5.5.2. Особенности работы элемента ToggleButton (выключатель) 256
5.5.3. Расчет стоимости авиабилетов с применением элементов ScrollBar и OptionButton 257
5.5.4. Расчет суммы чисел с применением элементов ScrollBar и SpinButton 259
5.5.5. Применение элемента SpinButton в задачах табулирования функции одной переменной 261
5.5.6. Обработка результатов тестирования 266
5.6. Управляющие элементы работы со вкладками MultiPage и TabStrip 268
5.6.1. Пример проекта с элементами MultiPage, ListBox, TextBox и Label 268
5.6.2. Организация меню на листе Excel для примера работы с массивом 273
5.6.3. Работа с набором вкладок TabStrip 276
5.7. Управляющий элемент ComboBox (поле со списком) 278
5.7.1. Проект перевода чисел в выбранную систему счисления – пример заполнения ComboBox значениями из массива 279
5.7.2. Проект «Сигналы светофоров» – пример заполнения ComboBox значениями из ячеек 280
5.7.3. Пример заполнения ComboBox методом AddItem 282
5.8. Применение технологий VBA для автоматизации задач АВС-анализа 283
5.9. Примеры работы с несколькими листами книги Excel 291
5.10. Пример разработки проектной документации 295
5.11. Примеры дополнительных элементов управления VBA 300
5.12. Решение графических задач в Excel и VBA 304
5.12.1. Построение линий в полярной системе координат 304
5.12.2. Применение ScrollBar для задач с построением диаграмм в Excel 309
5.12.3. Построение диаграмм средствами VBA 310
5.12.4. Работа с графическими объектами 317
5.13. Краткие рекомендации по приемам программирования в VBA 319
6. ОСНОВЫ ПРОГРАММИРОВАНИЯ В СРЕДЕ DELPHI 332
6.1. Интегрированная среда разработки Delphi 10.3 332
6.1.1. Состав главного окна Delphi 10.3 332
6.1.2. Особенности создания проектов в среде Delphi 334
6.1.3. Способы ввода и вывода данных 339
6.2. Программирование алгоритмической структуры следования 341
6.2.1. Примеры разработки приложений в Delphi 341
6.2.2. Технологии использования компонента Image 347
6.3. Управляющие структуры языка Delphi 350
6.3.1. Программирование ветвлений на Delphi 351
6.3.2. Переключатели Delphi для множественного выбора 355
6.4. Циклы и массивы в Delphi 357
6.4.1. Табулирование функции в таблице StringGrid 357
6.4.2. Основы работы с одномерными массивами 361
6.4.3. Двумерные массивы в среде Delphi 366

6.5.Процедуры и функции Delphi: создание и вызов в программе 367
6.6.Примеры графических приложений в Delphi 370
6.6.1. Отображение графики на канве Canvas 371
6.6.2. Построение диаграмм с помощью компонента Chart 375
6.7.Контрольные вопросы 381
7. ТЕМАТИКА ЗАДАНИЙ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ 383
МОДУЛЬ «СТРУКТУРНОЕ ПРОГРАММИРОВАНИЕ» 383
7.1.Задачи по теме «Основы алгоритмизации» 383
7.2.Задачи по теме «Линейный вычислительный процесс: технологии ввода и вывода данных» 388
7.3.Задачи по теме «Программирование алгоритмов линейной структуры с применением управляющих элементов» 390
7.4.Задачи по теме «Разветвляющийся вычислительный процесс» 392
7.5.Задачи по теме «Вычисляемый цикл» 395
7.6.Задачи по теме «Циклы с условием» 395
7.7.Задачи по теме «Табулирование функции одной переменной» 396
7.8.Задачи по теме «Одномерные массивы на листе Excel» 398
7.9.Задачи по теме «Одномерные массивы на пользовательской форме» 398
7.10. Задачи с блок-схемами алгоритмов обработки массивов 400
7.11. Задания на массивы со смысловым содержанием 404
7.12. Задачи по теме «Обработка двумерных массивов на листе Excel» 406
7.13. Задачи по теме «Обработка двумерных массивов на пользовательской форме» 406
МОДУЛЬ «ЧИСЛЕННЫЕ МЕТОДЫ» 407
7.14. Задачи по теме «Решение нелинейных уравнений» 407
7.15. Задачи по теме «Метод золотого сечения для поиска экстремума» 408
7.16. Задачи по теме «Вычисление значения определенного интеграла» 408
7.17. Задачи по теме «Вычисление суммы членов ряда» 410
МОДУЛЬ «ИНЖЕНЕРНО-ЭКОНОМИЧЕСКИЕ РАСЧЕТЫ» 411
7.18. Задачи на обработку массивов с применением элемента ListBox 411
7.19. Задание на разработку модели теста «Этика бизнеса» 411
7.20. Задачи по теме «Записи» 413
МОДУЛЬ «РАБОТА С ТЕКСТОМ И ГРАФИКОЙ» 413
7.21. Задачи по теме «Обработка строковых переменных» 413
7.22. Задачи по теме «Управление экраном и звуком» 414
7.23. Задачи по теме «Графика» 416
Вопросы и задания для самопроверки 417
Заключение 421
Список литературы 422

Введение

Широкое распространение в настоящее время получили среды быстрой разработки приложений (RAD – Rapid Application Development), позволяющие пользователям сократить объем создаваемого вручную кода.

Описанию примеров реализации базовых алгоритмов в трех средах визуальной разработки – на диалектах языка Pascal, выросших далее в Delphi, и в VBA посвящено данное издание.

Вначале рассматриваются теоретические вопросы алгоритмизации и особенности программирования на известных реализациях языка Pascal. Значительный объем практикума отведен изучению VBA для Excel – прекрасной среды начального обучения программированию. VBA – это событийно-ориентированный язык программирования, позволяющий расширять приложения Office. Далее показана работа в среде быстрой разработки приложений Delphi, которая является одной из наиболее совершенных систем визуального программирования.

Изучение основ офисного программирования в системе VBA Excel и знакомство с визуальной средой быстрой разработки Windows-приложений Borland Delphi – это возможность создания полнофункциональных профессиональных проектов на базе современной объектно-ориентированной концепции.

В главе 1 «Алгоритмизация и технологии программирования» приведены определение, свойства и способы представления алгоритма, этапы решения задач на ЭВМ, технологии программирования, описаны эволюция и направления развития языков программирования, этапы работы с программами, базовые алгоритмические структуры.

Глава 2 «Основы программирования на языке Pascal» посвящена вопросам представления информации и выполнения арифметических операций в различных системах счисления, изучению основ машинной арифметики, примерам реализации базовых структур алгоритмов в различных диалектах языка Pascal.

Глава 3 «Основы программирования на VBA» содержит описание объектной модели Excel, состава интегрированной среды разработки, основных конструкций языка, технологий ввода и вывода данных на VBA.

Наибольший объем публикации занимают глава 4 «Практикум по основным алгоритмическим конструкциям VBA» и глава 5 «Разработка графического интерфейса проекта в VBA».

Глава 4 предназначена для изучения методологии структурного программирования, сделан упор на создание макросов, работу с базовыми структурами, технологии обработки массивов и данных пользовательского типа, пользовательские процедуры и функции.

Глава 5 демонстрирует примеры разработанных пользовательских приложений на языке VBA в Excel. Рассмотрено размещение на листе рабочей книги элементов управления формы и элементов управления ActiveX. Представлены процедуры численных методов решения нелинейных уравнений, полиномов, вычисления интегралов и суммы ряда, нахождения экстремумов. Подробно разо-

браны примеры рекурсии – схема Горнера, вычисление факториала, приближенное нахождение суммы бесконечных рядов; методы численного интегрирования – методы прямоугольников, трапеций, парабол. Изложены приемы разработки графического интерфейса проекта, приемы построения диаграмм и объектов векторной графики VBA. Приведены примеры применения технологий VBA для автоматизации задач АВС-анализа.

При работе с пользовательской формой показаны многочисленные примеры процедур, использующих стандартные элементы управления, включенные в VBA: *CommandButton*, *Label*, *TextBox, ComboBox, ListBox, CheckBox, OptionButton, Image, Frame, ToggleButton, SpinButton, ScrollBar, TabStrip и MultiPage.*

В главе 6 «Основы программирования в среде Delphi» рассмотрены интегрированная среда разработки и особенности создания проектов в Delphi 10.3, примеры приложений табулирования функции, обработки массивов, использования графических возможностей Delphi, технологии использования процедур и функции, компонентов *Image*, *StringGrid*, *Chart*.

Глава 7 «Тематика заданий для самостоятельной работы» содержит более 250 задач, сгруппированных в модули «Структурное программирование», «Численные методы», «Инженерно-экономические расчеты», «Работа с текстом и графикой».

Структурированность данного издания, возможность изучения более 200 подробно разобранных примеров базовых алгоритмов в разных средах разработки способствуют усвоению и закреплению пройденного материала, проверке знаний.

Практикум предназначен для тех, кто изучает языки программирования самостоятельно или занимается обучением в любой среде программирования. Он может быть полезен программистам, математикам, экономистам, студентам и преподавателям.

В результате изучения данного издания и начинающие программисты, и пользователи, принявшие решение автоматизировать часть рутинной работы в Excel, смогут освоить технологии создания и редактирования макросов, визуального проектирования и событийного программирования – технологии конструирования диалоговых окон, проектирования графического пользовательского интерфейса, построения объектно-ориентированных конструкций и создания интегрированных с офисными программами приложений.

1. АЛГОРИТМИЗАЦИЯ И ТЕХНОЛОГИИ ПРОГРАММИРОВАНИЯ

1.1. Определение алгоритма и его свойства

Понятие алгоритма является фундаментальной категорией математики. В Толковом словаре по информатике (1991 г.) дано общепринятое понятие: ***алгоритм*** – точное предписание, определяющее вычислительный процесс, ведущий от варьируемых начальных данных к искомому результату. Алгоритм позволяет посредством конечного числа шагов получить решение задачи, однозначно определяемое исходными данными. Алгоритм – это строгая, четкая конечная последовательность математических и логических операций, приводящая к решению задачи.

Алгоритмизация процессов в широком смысле – это описание процессов на языке математических символов для получения алгоритма, отображающего элементарные акты процесса, их последовательность и взаимосвязь. В более узком смысле ***алгоритмизация*** – это процедура поиска, разработки и описания алгоритма решения задачи.

Свойства алгоритма

При составлении и записи алгоритма необходимо, чтобы он обладал следующими свойствами:

- детерминированность,
- массовость,
- результативность,
- дискретность,
- конечность,
- корректность.

Детерминированность (определенность, точность, однозначность) заключается в том, что при задании одних и тех же исходных данных несколько раз алгоритм будет выполняться абсолютно одинаково, всегда будет получен один и тот же результат. На каждом шаге выполнения алгоритма всегда точно известно, что делать дальше, каждое действие однозначно понятно исполнителю и не может быть истолковано неопределенно. Благодаря этому свойству выполнение алгоритма носит механический характер.

Массовость выражается в том, что с помощью алгоритма можно решать не одну конкретную задачу, а любую задачу из некоторого класса однотипных задач при всех допустимых значениях исходных данных.

Результативность (направленность) означает, что выполнение алгоритма обязательно должно привести к решению поставленной задачи, либо к сообщению о том, что при заданных исходных величинах задачу решить невозможно. Алгоритмический процесс не может обрываться безрезультатно.

Дискретность означает, что алгоритм состоит из последовательности отдельных шагов – элементарных действий, выполнение которых не представляет

сложности. Именно благодаря этому свойству алгоритм может быть реализован на ЭВМ.

Конечность (финишность) заключается в том, что последовательность элементарных действий алгоритма не может быть бесконечной, неограниченной, хотя может быть очень большой, например, если требуется большая точность вычислений.

Корректность означает, что если алгоритм создан для решения определенной задачи, то для всех допустимых значениях исходных данных он должен всегда давать правильный результат, ни для каких допустимых исходных данных не будет получен неправильный результат.

Если разработанная последовательность действий не обладает хотя бы одним из перечисленных выше свойств, то она не может считаться алгоритмом.

1.2. Способы записи и средства описания алгоритма

Описание алгоритма вполне допустимо ***на естественном языке***, но в естественных языках не всегда форме конкретного предложения соответствует единственное содержание. Так, появилась идея построения искусственных формальных языков, в результате чего возникло множество алгоритмических языков.

Алгоритмический язык – это система обозначений формальной записи алгоритмов, предназначенных для некоторого исполнителя. Алгоритмический язык довольно близок к обычному разговорному, но более точный, конкретный, лаконичный. В его состав входят операторы, команды, служебные слова и служебные символы. Алгоритмический язык имеет свой синтаксис и семантику.

Графическое представление алгоритма в виде структурной схемы – ***блок-схема*** – помогает сделать описание алгоритма наглядным. Укрупненная схема позволяет видеть функциональные связи между отдельными фрагментами алгоритма, а более подробные схемы содержат детали, составляющие содержание этих фрагментов.

Для реализации на ПК алгоритм необходимо описать на одном из языков программирования в виде программы. При вводе в ПК специальная программа-транслятор "переводит" алгоритм на машинный алгоритмический язык, в котором все данные и все действия представляются, в конечном счете, в виде двоичных чисел. После команды на запуск программа выполняется автоматически.

Способы представления алгоритма:

- словесно-формульный способ;
- табличный;
- графический способ;
- псевдокоды;
- программа на алгоритмическом языке.

Первые способы представления алгоритма служат для понимания решения задачи самим человеком, последний способ – единственный «понятный» компьютеру как автоматическому исполнителю.

Словесно-формульный способ записи алгоритмов представляет собой описание последовательных этапов обработки данных. Алгоритм задается в произвольном изложении на естественном языке.

Табличный способ представления алгоритма встречается в расчетных книжках при плате за квартиру, в бухгалтерских ведомостях, в таблицах инженерных расчетов и др.

Псевдокод представляет собой систему обозначений и правил для единообразной записи алгоритмов, он близок к обычному естественному языку. Использование в псевдокоде формальных конструкций и математической символики облегчает переход от записи на псевдокоде к записи алгоритма на формальном языке. Служебные слова выделяются в печатном тексте жирным шрифтом, а в рукописном тексте подчеркиваются.

Общий вид алгоритма, представленного псевдокодом:

алг название алгоритма (аргументы и результаты)
дано условия применимости алгоритма
надо цель выполнения алгоритма
нач описание промежуточных величин
последовательность команд (тело алгоритма)
кон

Часть алгоритма от слова **алг** до слова **нач** называется ***заголовком***, а часть, заключенная между словами **нач** и **кон** – ***телом алгоритма***. В предложении **алг** после названия алгоритма в круглых скобках указываются характеристики (**арг**, **рез**) и тип значения (**цел**, **вещ**, **сим**, **лит** или **лог**) всех входных (аргументы) и выходных (результаты) переменных. При описании массивов (таблиц) используется служебное слово **таб**, дополненное граничными парами по каждому индексу элементов массива.

Пример записи на псевдокоде алгоритма вычисления суммы квадратов целых чисел от 1 до n.

алг Сумма квадратов (**арг цел** n, **рез цел** S)
дано | $n > 0$
надо | $S = 1*1 + 2*2 + 3*3 + ... + n*n$
нач цел i
ввод n
S=0
нц для i **от** 1 **до** n
S=S+i*i
кц
вывод "S = ", S
кон

Программа – изложение алгоритма специально для ЭВМ в понятных ей символах, словах и командах (иначе говоря – языком программирования).

Для разработки структуры программы удобнее пользоваться записью алгоритма в виде ***блок-схемы*** (в англоязычной литературе используется термин

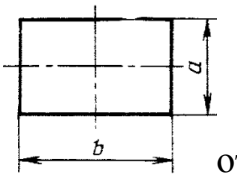
flow-chart). Для изображения основных алгоритмических структур и блоков на блок-схемах используют специальные графические символы (рис. 1.2.1).

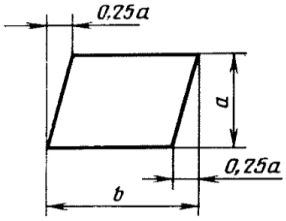
Название блока	Обозначение	Название блока	Обозначение
Начало или конец алгоритма	начало	Решение	нет X<=3 да
Процесс (действие или серия действий)	Y= X² + 5	Предопределенный процесс (вспомогательный алгоритм)	Расчет параметров
Ввод/вывод данных	ввод X	Подготовка	I=1, n
Линии потока		Комментарии	

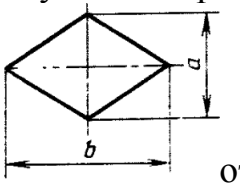
Рис. 1.2.1. Условное обозначение и примеры заполнения символов блок-схемы

Размеры и соотношения размеров фигур приводятся в ГОСТ 19–003–80 [1]. Согласно им все размеры связаны с двумя величинами *а* и *в*, где *а* – величина из ряда 10, 15, 20 мм. Допускается увеличивать размер *а* на число, кратное 5. Размер *в* равен 1,5*а*, допускается устанавливать *в*, равным 2*а*.

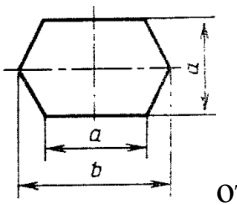
Описание основных символов схемы данных:

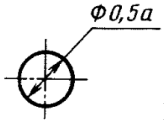
- символ «Процесс» 

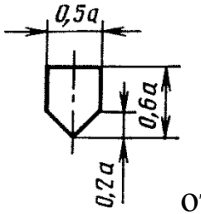
отображает функцию обработки данных любого вида, например, в него может быть вписана расчетная зависимость (формула);
- символ «Ввод – вывод» 

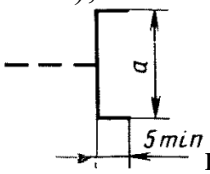
отображает преобразование данных в форму, пригодную для обработки или отображения результатов;
- символ «Решение» 

отображает решение или функцию переключательного типа, имеющую один вход и ряд альтернативных выходов, один и только один из которых может быть активирован после вычисления условий, определенных внутри этого символа (соответствующие результаты вычисления могут быть записаны по соседству с линиями, отображающими эти пути);



- символ «Подготовка» отображает модификацию команды или группы команд с целью воздействия на некоторую последующую функцию;
- символ «Линия потока» ——— отображает поток данных или управления, при необходимости или для повышения удобочитаемости могут быть добавлены стрелки – указатели для неестественных направлений потока справа-налево и снизу вверх;
- специальные символы «Соединитель» 

и «Межстраничный соединитель» 

отображают вход в часть схемы и выход из другой части этой схемы, используются для обрыва линии и продолжения ее в другом месте (соответствующие символы – соединители должны содержать одно и то же уникальное обозначение);
- специальный символ «Комментарий» 

используют для добавления описательных комментариев или пояснительных записей в целях объяснения или примечаний [2].

1.3. Рисование блок-схемы алгоритма в приложениях Microsoft Office

Для рисования элементов блок-схемы на вкладке *Вставка* в группе *Иллюстрации* следует нажать кнопку *Фигуры*, выбрать раздел «Блок-схема», а затем щелкнуть и нарисовать нужную фигуру.

Чтобы при рисовании автофигур приравнять их к границам ячеек листа Excel, необходимо рисовать или перемещать автофигуру при нажатой клавише *<Alt>*.

Между фигурами можно добавить прямые линии и линии с уступом в разделе «Линии». Соединительные линии содержат точки соединения для прикрепления линии к фигуре (рис. 1.3.1).

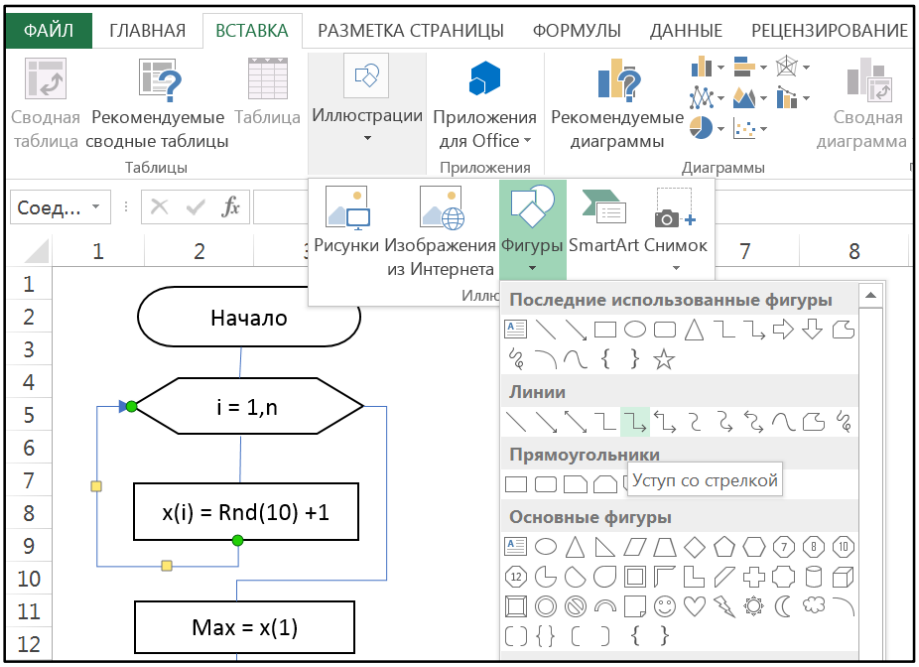


Рис. 1.3.1. Добавление соединительных линий

Для многократного добавления фигуры применяется команда «Зафиксировать режим рисования» в контекстном меню, вызываемом щелчком правой клавиши мыши по выбранной фигуре (рис. 1.3.2).

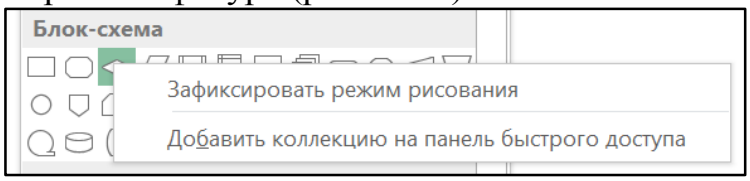


*Рис. 1.3.2. Команда для м*ногократного добавления фигуры

Для изменения типа и цвета соединительных линий на вкладке *Формат* в группе *Стили фигур* используются возможности кнопки *Контур фигуры*, например, направление стрелки (рис. 1.3.3).

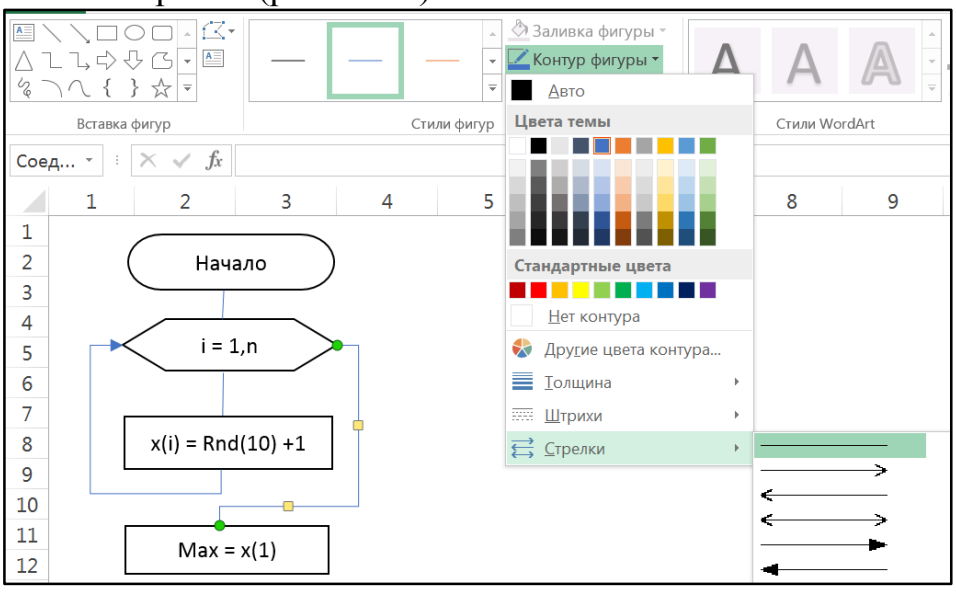


Рис. 1.3.3. Возможности кнопки Контур фигуры

Ввод текста осуществляется непосредственно на фигуре блок-схемы, либо для ввода текста следует щелкнуть нужную фигуру правой кнопкой мыши, выбрать команду «Изменить текст» и ввести текст.

Для размещения текста рядом с линиями или соединительными линиями используются надписи.

1.4. Этапы решения прикладных задач на компьютере

Программирование (programming) – теоретическая и практическая деятельность, связанная с созданием программ. Решение задач на компьютере включает в себя этапы, которые необходимо пройти для достижения конечной цели – решить задачу. Часть этапов осуществляется без участия компьютера.

Первый этап – ***постановка задачи*:**

• сбор информации о задаче;
• формулировка условия задачи;
• определение конечных целей решения задачи;
• определение формы выдачи результатов;
• описание данных (их типов, диапазонов величин, структуры).

На этом этапе участвует разработчик, который хорошо представляет предметную область задачи. Он должен четко определить цель задачи, выбрать необходимый объем исходной информации, привести описание каждого исходного данного и указать место его хранения, дать словесное описание содержания задачи и предложить общий подход к ее решению.

Второй этап – ***математическое описание задачи, анализ и исследование задачи, модели*:**

• анализ существующих аналогов;
• анализ технических и программных средств;
• разработка математической модели;
• разработка структур данных.

Этот этап выполняет разработчик, способный разработать математическое описание задачи. Цель этого этапа – создать математическую модель решаемой задачи, которая может быть реализована в компьютере.

Третий этап – ***разработка алгоритма:***

• выбор метода проектирования алгоритма;
• выбор формы записи алгоритма (блок-схемы, псевдокод и др.);
• выбор тестов и метода тестирования;
• проектирование алгоритма.

На основе математического описания необходимо разработать алгоритм решения. Чаще всего алгоритм изображается в виде блок-схемы с четко определенной последовательностью действий. Алгоритм должен быть понятным любому пользователю и пригодным для решения схожих с поставленной задач. На этом этапе алгоритмист использует принципы структурного программирования – идет по пути нисходящего проектирования, т.е. разрабатывает алгоритм на модульной основе.

Четвертый этап – ***программирование задачи***:

• выбор языка программирования;

• уточнение способов организации данных;

• запись алгоритма на выбранном языке программирования;

• возможно уточнение алгоритма – введение новых блоков, замена одних блоков на другие и т.д.

Этот этап выполняет разработчик, умеющий программировать. Программа – это представление алгоритма с помощью специальных символов, воспринимаемых компьютером. Составление программы обеспечивает возможность физической реализации алгоритма и соответственно поставленной задачи.

Пятый этап – ***тестирование программы:***

• синтаксическая отладка;

• отладка семантики и логической структуры;

• тестовые расчеты и анализ результатов тестирования;

• совершенствование программы.

Чтобы убедиться в правильности составленной программы, необходимо разработать тестовую задачу, которая позволит осуществить проверку всех ветвей алгоритма. Тест или контрольный пример – это совокупность таких исходных данных, на основании которых заранее определяются значения выходных данных. Тестовую задачу разрабатывает программист.

Шестой этап – ***перенос программы на машинный носитель и отладка программы.*** При работе на ПК программа и исходные данные вводятся с клавиатуры в оперативную память компьютера. Результаты решения сравниваются с полученными на пятом этапе расчетными значениями. По результатам сравнения делается заключение, что программа работает правильно, если результаты совпадают. В противном случае в программе есть ошибки.

Седьмой этап – ***получение и анализ результатов***. После устранения всех ошибок, выявленных тестовой задачей, можно перейти к получению результатов решения поставленной задачи. Подготавливаются исходные данные этой задачи и вводятся в компьютер. Полученные в результате решения выходные данные анализируются постановщиком задачи. На основании этого анализа вырабатываются решения, рекомендации, выводы. Оформляется техническая документация на разработанную программу.

Восьмой этап – ***сопровождение программы:***

• доработка программы для решения конкретных задач;

• составление документации к решенной задаче, к математической модели, к алгоритму, к программе, к набору тестов, к использованию программы.

Рассмотрим этапы математической постановки задачи подробнее:

a) ***обозначение переменных*** – лучше давать переменным название, отражающее их смысл;
b) классификация переменных на ***исходные данные и результат***, для исключения повторных расчетов вводятся ***промежуточные данные***;

c) ***классификация переменных по видам и по типам*** – простые и индексированные (структурированные);
d) ***расчетные формулы***, поясняющие каким образом из исходных данных могут быть получены результаты. Расчетные формулы записываются в порядке поступления в память компьютера и в виде, пригодном для реализации на ЭВМ. Желательно также определить ограничения, накладываемые на значения, допустимые и недопустимые операции по отношению к различным типам исходных данных.

Исходные данные бывают постоянные, условно-постоянные и переменные. Постоянные данные сохраняют свои значения в процессе решения задачи и не зависят от внешних факторов. Переменные данные изменяют свои значения в процессе решения задачи. Изменение значения условно-постоянных данных зависит не от процесса решения задачи, а определяются внешними факторами (количество дней в году, курсы валют, величина налога).

По структурному признаку данные классифицируют на простые (числовые и нечисловые) и структурированные (однородные и неоднородные).

Связь переменных с памятью компьютера. Простая переменная занимает одну ячейку памяти компьютера, для структурированных данных применимо одно имя для многих значений. Все элементы однородной структурированной переменной, например, вектора или массива однотипны. Для представления неоднородных структур используют запись, поименованные поля в которой содержат значения определенного типа.

Связь математической постановки задачи и блок-схемы алгоритма. Результат математической постановки задачи можно сравнить с блок-схемой алгоритма решения задачи – между вводом исходных данных и выводом результатов в блок-схеме расположены расчетные формулы описанного в математической постановки задачи алгоритма. Так можно контролировать правильность написания и математической постановки задачи, и блок-схемы алгоритма. В случае соответствия математической постановки задачи и ее блок-схемы можно переходить к этапу составления программы задачи.

1.5. Жизненный цикл программного продукта

Все программы по характеру использования можно разбить на два класса:

- утилитарные программы – для удовлетворения нужд их разработчиков, программы «для себя»;
- программные продукты – для удовлетворения потребностей пользователей, широкого распространения и продажи.

Программный продукт должен быть соответствующим образом подготовлен к эксплуатации, иметь необходимую техническую документацию, предоставлять сервис и гарантию надежной работы программы, иметь товарный знак

изготовителя. Только при таких условиях созданный программный комплекс может быть назван ***программным продуктом*.**

Программный продукт имеет несколько качественных характеристик:

- алгоритмическая сложность;
- полнота функций обработки;
- объем файлов программ;
- требования к операционной системе и техническим средствам обработки со стороны программного средства;
- объем дисковой памяти;
- размер оперативной памяти.

В условиях существования рынка программных продуктов важными характеристиками являются стоимость, количество продаж, время нахождения на рынке, известность фирмы-производителя и самой программы, наличие на рынке программных продуктов аналогичного назначения.

Программный продукт любого вида характеризуется жизненным циклом.

Жизненный цикл создания и использования компьютерных программ отражает различные их состояния, начиная с момента возникновения необходимости в данном программном изделии и заканчивая моментом его полного выхода из употребления у всех пользователей. Традиционно выделяют следующие основные этапы жизненного цикла программного обеспечения: анализ требований; проектирование; кодирование (программирование); тестирование и отладка; эксплуатация и сопровождение.

Выдвинутые в процессе эксплуатации программы идеи обычно используют после окончания жизненного цикла при разработке последующего, более совершенного программного продукта. Жизненный цикл информационных продуктов и услуг составляет основу жизненного цикла информационных технологий и, соответственно, информационных систем.

Требования, регламентирующие разработку, сопровождение, изготовление и эксплуатацию программ указаны в стандартах ***единой системы программной документации (ЕСПД)*** – комплексе государственных стандартов Российской Федерации, устанавливающих взаимосвязанные правила разработки, оформления и обращения программ и программной документации. Помимо государственных стандартов действуют отраслевые стандарты, стандарты предприятий.

ЕСПД обеспечивает возможность унификации программных изделий для взаимного обмена программами и применения ранее разработанных программ в новых разработках, снижения трудоемкости и повышения эффективности разработки, сопровождения, изготовления и эксплуатации программных изделий, автоматизации изготовления и хранения программной документации.

В области программирования общепризнаны стандарты института ANSI (Американский национальный институт стандартов) и Международной организации по стандартизации, ИСО (International Organization for Standardization, ISO).

1.6. Технологии и языки программирования

Технологии программирования – это апробированные стратегии создания программ в виде методик с описаниями алгоритма (проектных процедур). Разработка рациональной стратегии конкретного проекта осуществляется на основе следующих технологий:

- структурного программирования;
- проектирования программ с рациональной структурой данных;
- объектно-ориентированного программирования;
- визуального программирования.

Структурное программирование – методология разработки программного обеспечения, которая появилась в 1970-х годах как следствие возрастания сложности решаемых на компьютерах задач. В основе методологии структурного программирования лежит представление программы в виде иерархической структуры блоков, любая программа строится из трех базовых управляющих структур: последовательность, ветвление, цикл; кроме того, используются подпрограммы. При этом разработка программы ведется пошагово, методом «сверху вниз».

Пошаговая детализация как метод проектирования алгоритмов

Технология нисходящего проектирования с пошаговой детализацией является неотъемлемой частью создания хорошо структурированных программ. При написании программы с использованием этой технологии вся задача рассматривается как единственное предложение (вершина), выражающее общее назначение программы. Так как вершина редко отображает достаточное количество деталей, на основании которых можно написать программу, то надо начинать процесс детализации. На первом уровне детализации вершина разделяется на ряд более мелких задач в том порядке, в котором эти задачи должны выполняться. Далее каждая из подзадач разбивается на подзадачи, принадлежащие второму уровню детализации.

Программист завершает процесс нисходящей разработки с пошаговой детализацией, когда алгоритм настолько детализирован, что его можно преобразовать в программу.

Достоинства метода пошаговой детализации:

- сохраняется концептуальная целостность программы: от сложного к простому;
- проектирование программы, кодирование, проверку и документирование можно делать параллельно;
- в каждый момент времени (даже в начале разработки) имеется работающий вариант программы;
- фразы естественного языка, будучи закомментированными, служат хорошим путеводителем по программе.

Рост размеров и сложности разрабатываемого ПО потребовал развития структурирования данных и разграничения доступа к глобальным данным программы, что привело к появлению и развитию ***технологии модульного программирования***. Модульное программирование предполагает выделение групп подпрограмм, использующих одни и те же глобальные данные, в отдельно компилируемые модули. Связи между модулями при использовании данной технологии осуществляются через специальный интерфейс. Данную технологию поддерживают современные версии языков Pascal и C (C++), языки Ада и Modula.

Структурное программирование стало основой всего, что сделано в методологии программирования, включая и объектное программирование. В основе концепции ***объектно-ориентированного программирования*** лежит понятие объекта – некой сущности, которая объединяет в себе поля (данные) и методы (выполняемые объектом действия).

Логическим продолжением объектно-ориентированного программирования являются наиболее популярные в настоящее время системы визуального программирования. ***Визуальное программирование*** – способ создания программ с использованием особой диалоговой оболочки путем манипулирования графическими объектами вместо написания кода в текстовом виде. Примеры визуальных языков программирования: UML, Simulink. Наиболее часто визуальное программирование используется для создания интерфейса программ.

Кроме перечисленных выделяют следующие виды программирования: функциональное, логическое, автоматное, процедурное, прототипное, аспектно-ориентированное, компонентно-ориентированное.

Событийно-ориентированное программирование – главная часть программы представляет собой один бесконечный цикл, который опрашивает Windows, следя за тем, не появилось ли новое сообщение о событии. При обнаружении события вызывается подпрограмма, ответственная за обработку события. Цикл продолжается, пока не будет получено сообщение «завершить работу». События могут быть пользовательскими (возникают в результате действий пользователя), системными (возникают в ОС, например, сообщение от таймера), программными (генерируются программой, например, для обработки ошибки).

Компонентный подход предполагает построение ПО из отдельных компонентов физически отдельно существующих частей ПО, которые взаимодействуют между собой через стандартизованные двоичные интерфейсы.

Отличительная черта современного этапа развития технологии программирования – создание и внедрение автоматизированных технологий разработки и сопровождения программного обеспечения – ***CASE-технологий*** (Computer-Aided Software/ System Engineering – разработка программного обеспечения программных систем с использованием компьютерной поддержки) путем автоматизации процессов анализа и интеграции поддерживающих средств.

Эволюция языков программирования

Под ***языком программирования*** понимают правила представления данных и записи алгоритмов их обработки, которые автоматически выполняются ЭВМ. Выделяют пять поколений языков программирования (табл. 1.6.1).

Машино-ориентированные языки, к которым относятся машинные языки, автокоды, языки символического кодирования, ассемблеры, требуют указания мелких деталей процесса обработки данных, близки к программированию непосредственно в машинных кодах используемого процессора. Первые программы имели простейшую структуру, они состояли из собственно программы на машинном языке и обрабатываемых ею данных. Программы на языке ассемблера используются при разработке минимального по объему программного обеспечения с максимальной производительностью. Так, сервисные программы, как правило, составлены на языках типа Ассемблер.

Таблица 1.6.1. Поколения языков программирования

Поколения	Языки программирования	Характеристика
Первое	Машинные	Ориентированы на использование в конкретной ЭВМ, сложны в освоении, требуют хорошего знания архитектуры ЭВМ
Второе	Ассемблеры, макроассемблеры	Более удобны для использования, но по-прежнему машинно-зависимы
Третье	Языки высокого уровня	Мобильные, человеко-ориентированные, проще в освоении
Четвертое	Непроцедурные, объектно-ориентированные, языки запросов, параллельные	Ориентированы на непрофессионального пользователя и на ЭВМ с параллельной архитектурой
Пятое	Языки искусственного интеллекта, экспертных систем и баз знаний, естественные языки	Ориентированы на повышение интеллектуального уровня ЭВМ и интерфейса с языками

Машинно-независимые языки – это средство описания алгоритмов решения задач и информации, подлежащей обработке. Они удобны в использовании для широкого круга пользователей и не требуют от них знания особенностей организации функционирования ЭВМ и вычислительной сети. Подобные языки получили название ***высокоуровневых*** языков программирования. Программы на таких языках представляют собой последовательности операторов, структурированных согласно правилам рассматриваемого языка, операторы языка описывают действия, которые должна выполнять система после трансляции программы на машинном языке.

Языки высокого уровня имитируют естественные языки, используя некоторые слова разговорного языка и общепринятые математические символы, что более удобно для пользователя. В настоящее время насчитывается свыше 2000 различных языков высокого уровня. Языки высокого уровня делятся на:

- ***объектно-ориентированные*** (Object Pascal, C++, Java, PHP, Ada, Delphi, Perl, SmallTalk, Simula, Actor), в основе которых лежит понятие объекта, сочетающего в себе данные и действия над нами;
- ***логические*** (Prolog, Lisp и др.), основанные на теории и аппарате математической логики и ориентированные на систематическое и формализованное описание задачи с тем, чтобы решение следовало из составленного описания;
- ***процедурные*** (императивные) (Basic, Pascal, Algol, Cobol, PL-1, Си, Ассемблер, Fortran, Модула-2, Рапира, REXX, Ada).

Логическое программирование – парадигма программирования, основанная на автоматическом доказательстве теорем, а также раздел дискретной математики, изучающий принципы логического вывода информации на основе заданных фактов и правил вывода. Логическое программирование основано на теории и аппарате математической логики с использованием математических принципов резолюций.

Процедурное (императивное) программирование является отражением фон Неймановской архитектуры компьютера. Написанная на процедурном языке программа – это последовательность команд, определяющих алгоритм решения задачи для преобразования содержимого памяти при его изменении от исходного к результирующему состоянию. Основным является оператор присваивания. Концепция памяти как хранилища значений, содержимое которого может обновляться операторами программы, является фундаментальной в императивном программировании. Используя процедурный язык, программист определяет языковые конструкции для выполнения последовательности алгоритмических шагов.

Языки ***четвертого*** поколения носят ярко выраженный ***непроцедурный*** характер, определяемый тем, что в программах формируются скорее соотношения, чем последовательности шагов выполнения алгоритмов. ***Непроцедурные (декларативные) языки*** составляют группу языков, описывающих организацию данных, обрабатываемых по фиксированным алгоритмам (табличные языки и генераторы отчетов), и языков связи с операционными системами.

Типичными примерами непроцедурных языков являются языки, используемые для задач искусственного интеллекта, например, Prolog, Langin.

Второй тенденцией развития языков программирования четвертого поколения являются ***поддерживающие объектно-ориентированные технологии языки***, базирующиеся на понятии программного объекта: Pascal, Basic, C++, SmallTalk, Simula и ряд других языков программирования.

Третьим направлением развития языков четвертого поколения можно считать ***языки запросов***, позволяющих пользователю получать информацию из баз

данных. Языки запросов имеют свой особый синтаксис. Среди языков запросов фактическим стандартом стал язык SQL (Structured Query Language).

Четвертым направлением развития являются языки ***параллельного программирования*** (модификация языка Fortran, языки Occam, SISAL, FP и др.), которые ориентированы на многомашинные, мультипроцессорные среды и др.

К ***пятому*** поколению относятся языки искусственного интеллекта, экспертных систем, баз знаний (InterLisp, ExpertLisp, IQLisp, SAIL и др.), и естественные языки, не требующие освоения какого-либо специального синтаксиса (Clout, Q&A, HAL и др.).

Направления развития языков программирования

В современной информатике можно выделить основные направления развития языков программирования: ***процедурное*** и ***непроцедурное*** (рис. 1.6.1).

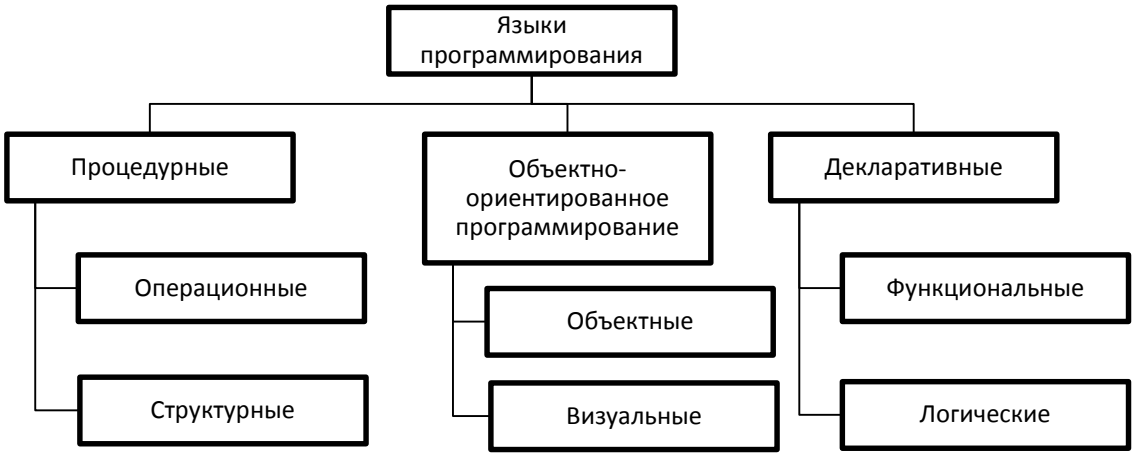


Рис. 1.6.1. Классификация языков программирования

В ***структурных*** языках (Pascal, Си, Ада, ПЛ/1) одним оператором записываются целые алгоритмические структуры: ветвления, циклы и т.д. В ***операционных*** языках (Фортран, Бейсик, Фокал) для этого используются несколько операций.

В ***функциональных языках*** программа описывает вычисление некоторой функции. Обычно функция задается как композиция других, более простых функций. Один из основных элементов в функциональных языках – ***рекурсия***, то есть вычисление значения функции через значение этой же функции от других элементов. Присваивания и циклов в классических функциональных языках нет. Тексты программ на функциональных языках программирования описывают «как решить задачу», но не *предписывают* последовательность действий для решения. Первым, спроектированным функциональным языком стал Лисп**.**

В ***логических языках*** программа задает данные и соотношения между ними. После этого системе можно задавать вопросы. Машина перебирает известные и заданные в программе данные и находит ответ на вопрос. Порядок перебора неявно задается самим языком. Классическим языком логического программирования считается Пролог.