Основы алгоритмизации и объектно-ориентированного программирования.

Алгоритм и его формальное исполнение

Алгоритм и его свойства.

Слово «алгоритм» происходит от латинского algorithmi – латинской формы написания имени математика IX века аль-Хорезми, который сформулировал правила выполнения арифметических операций.

Алгоритм – это строго определенная последовательность действий, описывающая процесс преобразования объекта из начального состояния в конечное, записанная с помощью понятных исполнителю команд.

В качестве объекта возьмем текст и построим алгоритм, описывающий процесс его редактирования.

Для того, чтобы изменить состояние объекта (*значения его свойств*), необходимо выполнить над ним определенную последовательность действий (*операций*). Выполняющий такие операции объект называется *исполнителем*.

Исполнителем редактирования текста может быть человек, компьютер и др.

Алгоритмы состоят из отдельных команд, которые исполнитель выполняет одну за другой в определенной последовательности.

Разделение информационного процесса в алгоритме на отдельные команды называется *дискретностью*.

Весь процесс редактирования текста должен быть разбит на отдельные операции, записанные в виде отдельных команд для исполнителя.

Каждый исполнитель обладает определенным набором, системой команд, которые он может выполнить. Алгоритм должен быть понятен исполнителю, т.е. содержать только те команды, которые входят в систему его команд.

Возможные операции в процессе редактирования текста: *удаление, копирование, перемещение или замена его фрагментов*. Исполнитель редактирования должен быть в состоянии выполнить все эти операции.

Запись алгоритма должна быть такова, чтобы, выполнив очередную команду, исполнитель точно знал. Какую команду необходимо исполнять следующей.

Это свойство алгоритма называется *детерминированностью*.

Должны быть определены начальное состояние объекта и его конечное состояние (цель преобразования). Алгоритм должен обеспечивать преобразование объекта из начального состояния в конечное за конечное число шагов. Такое свойство алгоритма называется *результативностью*.

Следовательно, для текста необходимо задать начальную последовательность символов и конечную, которая должна быть получена после редактирования.

Формальное выполнение алгоритма.

Алгоритм позволяет *формализовать* выполнение информационного процесса, т. е. не вникать в содержание поставленной задачи, а только строго выполнять последовательность действий, предусмотренную алгоритмом.

Рассмотрим редактирование текста.

Наш объект – фрагмент.

Нужно: перевести его из исходного состояния в конечное – поменять местами слова во фразе «информационная модель».

Запишем алгоритм на *естественном языке*, т.е. понятном пользователю:

1. Выделить слово «информационная» + пробел.
2. Вырезать этот фрагмент и поместить его в буфер обмена.
3. Установить курсор на позицию после слова «модель» + пробел.
4. Вставить вырезанный фрагмент текста.

Формальная модель представляет текст делящимся на страницы, состоящие из определенного количества строк, которые в свою очередь, включают определенное количество знакомест (символов).

На формальном языке алгоритм следующий:

1. Выделить символы с 1 по 15.
2. Вырезать этот фрагмент и поместить его в буфер обмена.
3. Установить курсор на позицию после 7-го символа.
4. Вставить выделенный фрагмент.

Фактически пользователь будет давать команды объектам программной среды Windows&Office, которые в свою очередь будут действительными исполнителями алгоритма.

Компьютер – автоматический исполнитель алгоритмов.

Представление информационного процесса в форме алгоритма позволяет поручить автоматическое исполнение различным техническим устройствам.

Алгоритм, записанный на «понятном» компьютеру языке программирования, называется *программой*.

Программа – последовательность команд.

Развитие языков программирования.

Информацию в компьютере обрабатывает процессор, следовательно, алгоритм должен быть записан на языке, «понятном» для процессора, т.е. на машинном языке, представляющем собой логические последовательности нулей и единиц.

В 50-е годы XX века, программы писались на машинном языке и представляли собой очень длинные последовательности нулей и единиц.

В 60 – 70-е годы для облегчения туда программистов начали создаваться *языки программирования высокого уровня*, формальные языки, кодирующие алгоритмы в привычном для человека виде (в виде предложений). Такие языки программирования строились на основе определенного алфавита и строгих правил построения предложений (синтаксиса).

Наиболее распространенным типом языков программирования высокого уровня являются *процедурные языки*. В таких языках широко используются управляющие конструкции (операторы), которые позволяют закодировать различные алгоритмические структуры (линейную, ветвление, цикл).

Одним из первых процедурных языков программирования был Бейсик (Basic), созданный в 1964 году. В течение последующего времени Бейсик развивался, появились его различные версии (MSX-Basic, Бейсик-Агат, QBasic и др.). Другим широко распространенным языком программирования алгоритмического типа является Pascal.

В настоящее время наибольшей популярностью пользуются системы объектно-ориентированного визуального программирования Microsoft Visual Basic и Borland Delphi. Для создания приложений в среде Windows&Office используется язык программирования Visual Basic for Applications (VBA).

Этапы разработки программы.

1. *Постановка задачи* – задача формулируется на естественном языке.
2. *Анализ, формализованное описание задачи, выбор модели*. Анализ – определение входных и выходных данных, выявление возможных ограничений на их значения. Формализованное описание задачи – предполагает ее математическую формулировку. Выбор модели – если речь идет о моделировании каких-либо явлений или процессов, на этом этапе разрабатывается математическая модель процесса (явления).

3. *Выбор или разработка алгоритма решения задачи*. Тщательно проработанный алгоритм решения задачи – необходимое условие эффективного программирования.
4. *Проектирование общей структуры программы* – «архитектурная» проработка проекта. Определяются те части алгоритма, которые целесообразно оформить в виде подпрограмм, модулей. Определяется способ хранения информации – в виде набора простых переменных, массивов, других структур. На этапе проектирования структуры программы обычно избегают привязки к особенностям конкретного языка программирования.

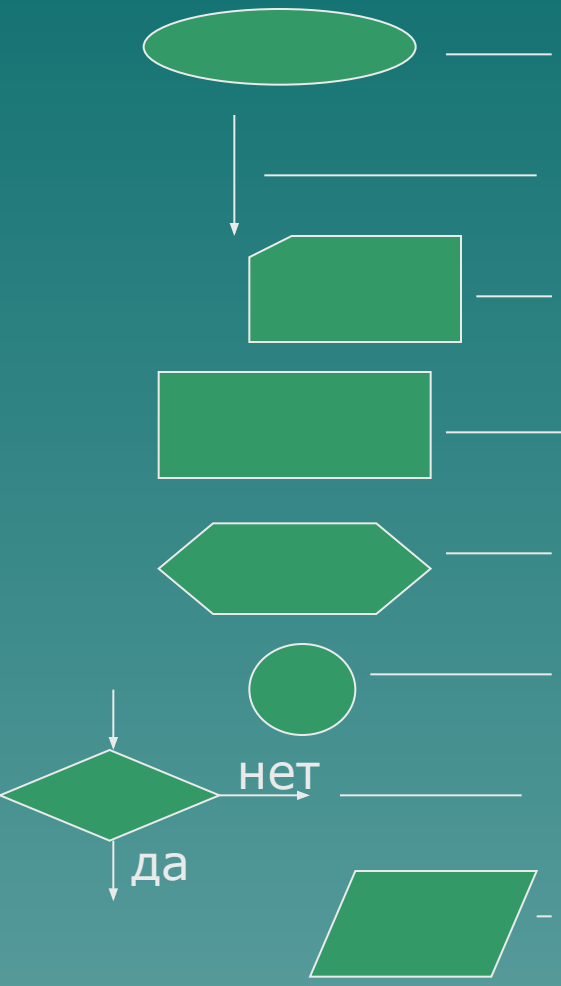
Одним из популярных подходов к проектированию программ является проектирование *«сверху вниз»*. В этом случае сначала определяется «глобальная» задача, которую должна решить программа (алгоритм). Затем эта задача разбивается на две, три или большее (но не очень большое) количество подзадач. Такое разбиение называется пошаговым уточнением процесса решения исходной задачи. Затем по отдельности рассматривается каждая из подзадач, которую в свою очередь, возможно, придется разбить на еще более мелкие подзадачи. При таком подходе программист может работать с небольшим количеством информации, что снижает риск упустить из виду какую-нибудь важную деталь.

5. *Кодирование* – это запись алгоритма на языке программирования.
6. *Отладка и верификация*. Отладка программы заключается в устранении ошибок программирования, ошибок перевода алгоритма на язык программирования. Верификация – это проверка того факта, что программа работает «правильно», т.е. дает правильный результат.

7. *Получение результата, его интерпретация с возможной последующей модификацией модели* – следует сравнить полученные с помощью компьютера результаты с результатами наблюдений. Процесс такого анализа и называется интерпретацией результатов расчета. Здесь программиста может ожидать разочарование – результат может отличаться от желаемого. В этом случае, возможно, придется изменить саму модель, сделав ее более реалистичной.

8. Публикация или передача заказчику результата работы.
9. Сопровождение программы – предполагает консультирование заказчика по работе программы, устранение замеченных в процессе ее эксплуатации недостатков (а возможно, и ошибок), обучение пользователей работе с программой.

Запись алгоритма в виде блок-схемы.



начало/конец алгоритма

передача управления

ввод данных

блок вычислений

начало (заголовок) цикла

конец цикла

ветвление

вывод данных