Тема 4. Инструментальное программное обеспечение

4.1. Основы алгоритмизации.
4.2. Языки программирования и системы программирования.
4.3. Программирование в среде офисных приложений.

Инструментальное программное обеспечение – программное обеспечение, предназначенное для использования в ходе проектирования, разработки и сопровождения программ.

Обычно этот термин применяется для акцентирования отличия данного класса ПО от прикладного и системного программного обеспечения.

4.1. Основы алгоритмизации.

Понятие алгоритма занимает центральное место в вычислительной математике и программировании. Справедливо следующее определение.

Алгоритм – описание последовательности действий, четкое выполнение которых приводит к решению поставленной задачи и получению результата.

Термин «алгоритм» (algorithmi) происходит от латинского написания имени узбекского математика и астронома IX века Мухаммеда ибн Муса аль-Хорезми, который в первые (825г.) разработал правила выполнения четырех арифметических действий в десятичной системе счисления.

Исполнитель алгоритма — человек (группа людей) или техническое устройство, которые понимают команды алгоритма и умеют правильно их выполнять.
У исполнителя есть своя система команд. Система команд исполнителя — команды, которые понимает и может выполнить исполнитель.
Компьютер является универсальным исполнителем.

Характерными свойствами алгоритма являются **определенность**, **массовость** и **результативность**.

Алгоритм обладает следующим набором основных свойств:
Определенность (детерминированность, точность) – свойство алгоритма, указывающее на то, что каждый шаг алгоритма должен быть строго определен и не допускать различных толкований;
Результативность – свойство, состоящее в том, что любой алгоритм должен завершаться за конечное (пусть даже очень большое) число шагов;
Массовость – применимость алгоритма ко всем задачам рассматриваемого типа, при любых исходных данных;

Дискретность (разрывность) – это свойство алгоритма, характеризующее его структуру: каждый алгоритм состоит из отдельных законченных действий, т. е. делится на шаги;
Формальность – это свойство указывает на то, что любой исполнитель, способный воспринимать и выполнять инструкции алгоритма, действует формально, т. е. отвлекается от содержания поставленной задачи и лишь строго выполняет инструкции.

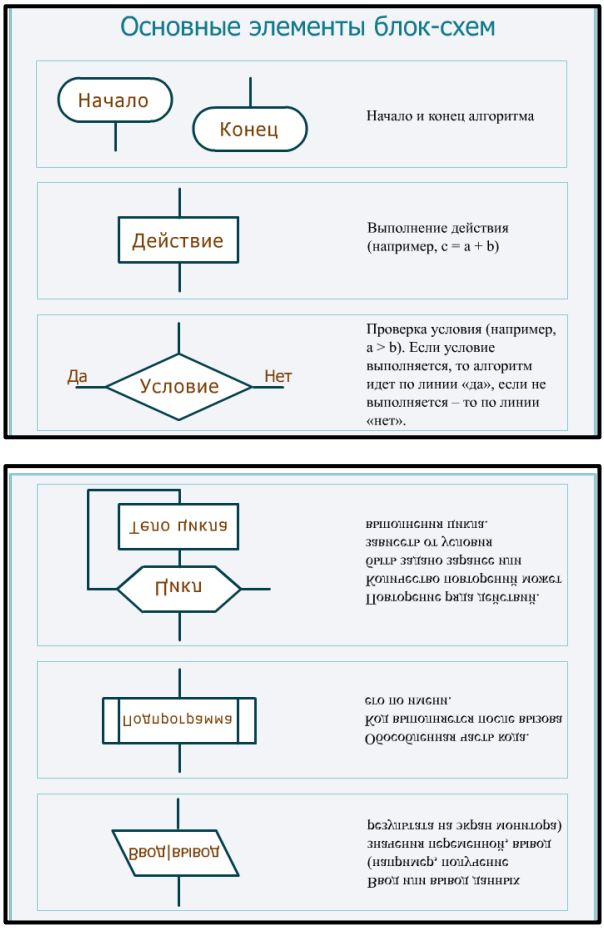
Существуют следующие **способы описания алгоритмов**:
1) запись на естественном языке (словесное описание);
2) изображение в виде схемы (графическое описание);
3) запись на алгоритмическом языке (составление программы).

Способы словесного описания алгоритмов отличаются применяемыми метаязыками (языки, предназначенные для описания языка программирования). Например, словесное описание алгоритма решения квадратного уравнения
$a \cdot x\ 2 + b \cdot x + c = 0$ будет выглядеть следующим образом:
1) D := b2 – 4 · a · c;
2) если D < 0, идти к 4;
3) x1 := (–b + D) / (2 · a); x2 := (–b – D) / (2 · a);
4) Останов.
Основной недостаток словесного описания – плохая наглядность.

Графическое описание алгоритма – это представление алгоритма в виде схемы (двухмерного рисунка), состоящей из последовательности блоков (геометрических фигур), каждый из которых отображает содержание очередного шага алгоритма

(управляющей структуры). Внутри фигур кратко записывают выполняемое действие. Такую схему называют блок-схемой алгоритма.

На рис. 1. приведены основные условные обозначения, используемые при графической записи алгоритма.



Для стандартизации и унификации языка схем алгоритмов в 1992 г. был принят ГОСТ 19.701–90 «Единая система программной документации. Схемы алгоритмов, программ, данных и систем. Условные обозначения и правила выполнения» [1]. В настоящее время данный стандарт продолжает действовать в Республике Беларусь.

Понятие структурного программирования В 70-е годы прошлого века возник новый подход к разработке алгоритмов и программ, который получил название структурного проектирования программ. К его достоинствам можно отнести: более высокую производительность; читаемость программ; простоту тестирования и эффективность программ. Одна из концепций структурного программирования – нисходящее проектирование. Это средство разбиения большой задачи на меньшие подзадачи так, чтобы каждую подзадачу можно было рассматривать независимо. Все операции в программе, построенной на основе структурного программирования, должны представлять собой либо исполняемые в линейном порядке выражения, либо одну из следующих управляющих конструкций: вызовы подпрограмм; вложенные на произвольную глубину операторы условия; циклические операторы.

Типы алгоритмов

Структурный подход к построению алгоритмов предполагает использование только нескольких основных структур, комбинации которых дают все многообразие алгоритмов. Все существующие алгоритмы делятся на три типа

линейный алгоритм при каждом исполнении предписывает однократное выполнение всех действий алгоритма в определенной последовательности;

разветвляющимся алгоритм **о**писывает несколько возможных последовательностей действий (ветвей алгоритма) и при каждом исполнении предписывает выполнение одной из ветвей алгоритма в зависимости от определенных условий;

циклический алгоритм при каждом исполнении предписывает многократное выполнение одной и той же последовательности действий.

В отличие от линейных алгоритмов, в которых команды выполняются последовательно одна за другой, в ветвящиеся алгоритмы входит условие, в зависимости от выполнения или невыполнения которого исполняется та или иная последовательность команд (или их серий). Условие составляется в виде **логического выражения** — сравнения некоторых объектов между собой (переменных или констант). Существует шесть операций сравнения:
• равно (=);
• не равно (<>);
• меньше (<);
• больше (>);
• меньше или равно (<=);
• больше или равно (>=).
Прочитать алгоритмы. примеры

Понятия алгоритма и программы не имеют четкого разграничения. Так, программа, записанная на алгоритмическом языке – это окончательный вариант алгоритма решения задачи, ориентированный на конкретного исполнителя (компьютер или язык программирования).

Программа - это алгоритм, записанный в виде последовательности команд, понятных ЭВМ (машинных команд). При записи алгоритмов в виде программ для ЭВМ используются языки программирования.

4.2. Языки программирования и системы программирования.

Системы программирования — совокупности специальных средств создания программ и их перевода на машинный язык для выполнения на ЭВМ.

Система программирования - это набор специализированных программных продуктов, которые являются инструментальными средствами разработчика. Программные продукты данного класса поддерживают все этапы процесса программирования, отладки и тестирования создаваемых программ.

Системы программирования включают в себя не только сам алгоритмический язык, определяющий синтаксис команд, но и их преобразователь в машинный код.

Эти средства служат для разработки новых программ.

- ❑ Системы программирования:

– редактор текста
– транслятор (компилятор, интерпретатор)
– библиотеки подпрограмм
– отладчики, для поиска ошибок в программе

- ❑ Языки программирования

Как любой человек разговаривает на определенном человеческом языке (может на одном, а может и на нескольких), так и ЭВМ способна понимать только «свой» машинный язык — набор команд, распознаваемых и выполняемых компьютером (точнее, процессором). Составленный алгоритм решения задачи следует перевести на понятный ЭВМ машинный язык, аналогично тому, как переводят обычные тексты на иностранные языки, например английский. Обычно процесс перевода состоит из двух частей.

Компилятор читает всю программу *целиком,* делает ее перевод и создает законченный вариант программы на машинном языке, который затем и выполняется.

Интерпретатор переводит и выполняет программу *строка за строкой.* Программа, обрабатываемая интерпретатором, должна заново переводиться на машинный язык при каждом ее очередном запуске.

Язык программирования — это искусственный формализованный язык со строго определенным синтаксисом для описания алгоритма решения задач на компьютере, который включает набор команд (операторов), правил и соглашений, более понятных пользователю, чем машинный язык.
По сложности языки программирования разделяют на:

- ✓ языки низкого уровня
- ✓ языки высокого уровня

Машинно-ориентированные языки:
машинный язык (язык машинных кодов)
ассемблер
Машинно-независимые языки:
Проблемно-ориентированные
-для решения специфических задач из некоторой отрасли знаний.

1. Фортран (Fortran)
2. Лисп (Lisp)
3. Кобол (Cobol)
4. Алгол (Algol)

Универсальные
-позволяют решить любую задачу, хотя трудоемкость в различных языках будет отличаться.

1. Бейсик (Basic)
2. Си (C)
3. C++
4. Паскаль (Pascal)

Объектно-ориентированные
современные среды визуального объектно-ориентированного программирования

1. Visual Basic
2. Delphi
3. Visual Fortran

Этапы разработки программного обеспечения
Разработка любого программного обеспечения включает четыре последовательных этапа:
• постановка задачи;
• моделирование задачи;
• алгоритмизация решения задачи;
• составление программы.
Технология программирования — это совокупность средств и методов создания программного обеспечения. Применение эффективных технологий программирования подразумевает:
• внедрение прогрессивных инструментальных средств разработки программ;

• использование специальных методов и приемов организации работ по разработке программ;
• стандартизированность, тиражируемость и воспроизведение методов программирования различными разработчиками.

Инструментарий технологии программирования — это программный комплекс, обеспечивающий технологию разработки, отладки и внедрения создаваемых программных продуктов. Он обеспечивает процесс разработки программ и включает специализированные инструментальные средства разработчика. **Пользователями технологии программирования** являются системные и прикладные программисты.

4.3. Программирование в среде офисных приложений.
Офисное программирование — это процесс разработки приложений, предназначенных для автоматизации офисной деятельности с использованием специализированных пакетов (MS Office, OpenOffice.org или подобных).

Офисное программирование как процесс разработки приложений имеет ряд особенностей по сравнению программированием как таковым. Эти особенности проявляются в таких аспектах, как:
цель разработки;
область применения;
язык программирования;
среда разработки;
поддержка объектно-ориентированного программирования.
Рассмотрим эти особенности на примере MS Office.
Цель разработки. В офисной среде программный проект неразрывно связан с документом, хранится как часть документа и не может существовать независимо от него. Документ, а не программа, является целью разработки.
Область применения офисного программирования весьма и весьма широка – от настройки отдельных документов до решения задач автоматизации офисной деятельности масштаба предприятия, в т.ч. ориентированных на совместную работу в глобальной сети.

Понятно, что основное назначение офисных приложений — автоматизация офисной деятельности, однако, средства офисного программирования применяют и для совсем других разработок, выходящих далеко за ее рамки. Для наглядности приведем ряд примеров: программа биоритмов, написанная на VBA для Excel; ASCII art на музыкальный клип AC/DC; макровирусы и множество других совсем не офисных приложений.

Единственный язык программирования, поддерживаемый пакетом MS Office является Visual Basic for Application (VBA). VBA — это инструмент разработки приложений, который позволяет создавать программные продукты, решающие практически все задачи, встречающиеся в среде MS Windows.

Среда разработки. Среда приложений Office ориентирована в первую очередь на пользователей, а не на программистов и в ней можно создавать документы без всякого программирования.

Среда MS Office предлагает два способа создания программ, отличающихся подходом к процессу: использование макрорекордера и написание исходного кода программ на языке VBA в интегрированной среде разработки. Эти подходы ориентированы на разные категории: непосредственно пользователей и программистов соответственно.

Макрорекордер (MacroRecorder) – это программный инструмент, записывающий действия пользователя при работе с документами и приложениями, с сохранением записи в виде макроса — исходного кода на языке VBA. При вызове сохраненного макроса воспроизводится вся сохраненная последовательность действий.

Интегрированная среда разработки на VBA (Visual Basic Environment, VBA) — встроенное в MS Office средство для написания, тестирования и отладки приложений на VBA.

Поддержка ООП. Разработка приложений для MS Office тесно связана с парадигмой объектно-ориентированного программирования. Все документы (более того, сами компоненты пакета) в MS Office — суть объекты, наделенные собственными наборами свойств (характеристик объекта), методов (подпрограмм управления свойствами) и событий (подпрограмм, обрабатывающих изменения состояния объекта в результате некоторых действий). Соответственно, для обеспечения более полной интеграции с пакетом, входной язык

(VBA) также поддерживает ООП. Все объекты приложения MS Office образуют иерархическую структуру, которая определяет связь между ними и способ доступа. Такая структура называется объектной моделью (object model).

✓ Visual Basic for Applications представляет общую языковую платформу для приложений Microsoft Office

✓ Позволяет повысить производительность выполнения многих команд в приложениях

✓ Достоинством языка Visual Basic for Applications является то, что среда разработки включает инструменты для визуального конструирования программ

В настоящее время VBA встроен:

1) во все главные приложения Microsoft Office:

– Word
– Excel
– Access
– PowerPoint
– Outlook
– FrontPage
– InfoPath

а также:

2) в приложения Microsoft:

– Visio
– Project

3) в более 100 приложений других фирм, например, в CorelDRAW и AutoCAD и т. п.

Это стандартное интерфейсное окно (Alt+F11), содержащее меню, панели инструментов, другие окна и элементы, которые применяются при создании проектов VBA. Общий вид окна редактора Visual Basic представлен на рис. 2.

Окно проекта (Project)

Окно свойств (Properties)

Окно просмотра объектов(Object Browser)

Окно Code (Окно редактирования кода)

Алфавит языка VBA

Все слова языка VBA можно разделить на четыре группы:

- имена
- ключевые слова

- числа
- строки

Переменная – это элемент данных в программе, которому присвоено имя. Значение переменной может задаваться и изменяться программой

Константа – в отличие от переменной константа никогда не меняет своего значения

Оператор присваивания

Операторы присваивания служат для изменения значения переменных:

ИмяПеременной = значение

S = 0

ИмяПеременной = выражение

S = S + 1

Процедуры VBA бывают двух типов:

1. общие процедуры
2. процедуры обработки событий

Процедуры могут быть:

1. процедурами-подпрограммами
2. процедурами-функциями

Пример

Вычислить значение функции $y = f(x)$, в точке x_0=5,5

$$y = \frac{e^{-x}}{\sqrt{e^{-x}+1}} - x$$

Вариант 1

```
1  Private Sub Пример1()
2  Dim x As Single
3  Dim y As Single
4  x = 5.5
5  y = Exp(-x) / Sqr(Exp(-x) + 1) - x
6  MsgBox "Значение Y=" & Str(y), , "Результат"
7  End Sub
```

Принципы построения процедуры

1. Объявление процедуры-подпрограммы с именем Пример1

2. Объявление переменной x и указание типа переменной Single (одинарное с плавающей точкой)
3. Объявление переменной y
4. Присвоение переменной x числового значения 5,5
5. Вычисление значения y (присвоение переменной y арифметического выражения)
6. Вывод результата в окно сообщения MsgBox
7. Окончание процедуры Пример1 (End Sub)

Вариант 2. (с окном ввода значения переменной)

```
Private Sub Пример2()
Dim x,y As Single
x = InputBox("Введите значение x:", _
"Ввод исходных данных")
y = Exp(-x)
y = y / Sqr(y + 1) – x
MsgBox "Значение Y=" & Str(y), , "Результат"
End Sub
```

Вариант 3. Назначение процедуры кнопки.
Вариант4. Создание функции пользователя.

Лекция обзорная, формирующая общее представления о возможностях VBA. Более углубленное изучение возможно в СНИЛ.