

Основы программирования: расширенный фундамент

(фрагмент)

Введение

В информационных технологиях существует много языков программирования. Многие из которых похожи друг на друга, и выучив один из них легко перейти к программированию на другом. Но не всегда — при изучении языка из другой группы требуется расширить понимание программирования. В данной работе рассмотрена три группы:

1. Средний уровень — императивный языки — в программе набор действий что нужно делать.
2. Низкий уровень — машинный код — программа похожа на то, как это на самом деле «крутится» в компьютере. Минимальная абстракция.
3. Высокий уровень — функциональные языки — программа состоит из функций, которые описывают решения задачи и подзадач и сами могут быть объектами манипуляции. Максимальная абстракция от деталей аппаратного обеспечения.

Чтобы программировать на базовом уровне, достаточно знать какой-либо язык (обычно это из «среднего уровня»). Привыкнув программировать на одном языке, программист может смутно представлять то как оно работает внутри или как это можно проще выразить, сократив трудозатраты и повысив надёжность. Для высококвалифицированного программиста это недопустимо.

Здесь приведены идеи по построению курса предварительной подготовки программистов, которые ориентируются на высокую квалификацию.

Обучение предполагается проводить по шагам. Каждый шаг построен с целью добиться понимания какого-либо отдельного понятия.

Курс не заменяет обычного изучения программирования, а подготавливает почву для него.

Все используемые языки программирования упрощены для лёгкого понимания (они не предназначены для написания настоящих программ). От обучаемых требуется воспроизведение и составление программ (от простых к сложным). (Если и так знают/быстро схватывают - можно пропускать и переходить к более сложному). Предлагается использование классной доски (или чего-либо её заменяющего), компьютер необязателен.

ЦА: умные (но ещё неопытные) люди, которые хотят научиться программировать (изначально с прицелом на продвинутый уровень)

Цель: подготовить фундамент для изучения разных подходов к программированию на [относительно] простых примерах.

Список изучаемых понятий:

Основные:

- . Алгоритм
- . Ветвление
- . Цикл
- . Функция
- . Текст

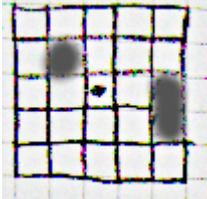
- . Блок-схема
- . Переменная

Дополнительные:

- . Входные/выходные данные
- . Состояние программы
- . Макрос
- . Синтаксическое дерево
- . Рекурсия
- . Единство кода и данных
- . Список/Массив
- . Компонент
- . Объектно-ориентированный подход
- . Обработка ошибок

Часть 1. Императивный подход.

Сначала нужно рассказать что такое алгоритм. Рабочая модель: робот на клетчатом поле (команды: **вверх, вниз, влево, вправо, закрасить_клетку**).



Программы представлены в двух основных вариациях: текст и блок-схема.

Шаг 1: Команды.

Рассказать про модель с роботом и индивидуальные команды, убедиться что обучаемый может проиллюстрировать результаты выполнения отдельных команд. Убедиться, что обучаемого не смущает, если преподаватель сам закрашивает/стирает клетки, перемещает робота и т.д. *Интеллектуальные операции: сравнение, обобщение — команды как единичное простое действие; предметное манипулирование — результаты выполнения команд; абстрагирование — отделение модели (движения робота) от сторонних факторов (нерассмотрение физического представления модели (рисунок на доске) или предполагаемых свойств настоящего робота)*

Шаг 2: Блок-схемы.

Оформить несколько команд в виде блок-схемы (вначале - два прямоугольника со стрелкой, никаких дополнительных существенных типа "начало программы", "конец программы" не нужно). Убедиться что обучаемый может выполнить цепочку команд.

Замечание: уже на этом этапе следует вызвать ситуации 1. закрашивание клетки, которая уже до этого закрашена (обучаемый должен закрасить её ещё раз (или хотя бы явно отметить что оно уже закрашено), 2. выход робота за границы поля (предполагается что обучаемый будет всё равно рисовать робота вне поля и всё равно закрашивать там клетки).

Далее убедиться что обучаемый может составить программы в виде блок-схем. Не забыть рассмотреть случаи: 1. когда программа состоит только из одной команды, 2. Когда программа

состоит из 0 команд - пустая.

Также убедиться, что обучаемый правильно реагирует (продолжает невозмутимо выполнять программу, даже если теперь она уже не приводит к поставленной цели), если в процессе воспроизведения его программы преподаватель меняет условия (закрашенность клеток, положение робота)

Диаграмма	До выполнения программы	«Вверх»	«Закрасить»	«Влево»

Интеллектуальные операции: программирование; синтез, планирование, кодирование, принятие решения — составление программ-блок-схем.

Шаг 3: Тексты

Оформить те же несколько команд, но в виде текста (одна команда на строке, полные названия команд, без сокращений). Убедиться что в виде такого текста программы тоже могут выполняться/составляться. Ввести дополнения в текст: сокращённые команды (в виде одних букв, в виде стрелочек, в виде специально выдуманных символов и (обязательно) в виде цифр), потом по несколько команд в одной строке. Убедиться, что это не вводит обучаемого в заблуждение и он может выполнять/составлять программы в разных формах (понимая, что это одна и та же программа). Показать пример, в котором помимо команд в тексте есть комментарии, которые не выполняются; показать пример, в котором к тексту предъявляются дополнительные требования (например, строки должны быть пронумерованы или в строке должно указываться количество команд). Напоследок можно придумать специальную вычурную, усложнённую форму (например, программа задаётся в виде последовательности разных узлов, завязанных на верёвочке)

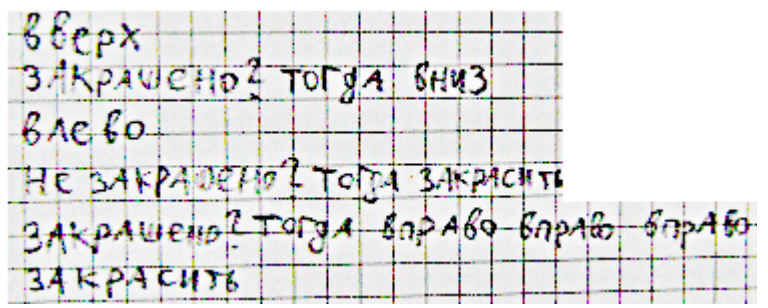
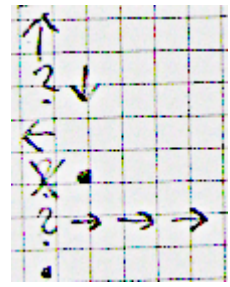
Разные варианты текста этой программы: слова	Разные варианты текста этой программы: выдуманные символы	Таблица соответствия разных форм команд из тех примеров.

буквы слова, но в одну строчку значки	цифры Усложнённая форма: перед каждым словом номер строки в угловых скобках, после каждого слова количество букв в нём после двоеточия, после программы надпись «~~конец».	
---	--	--

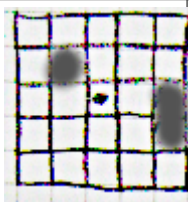
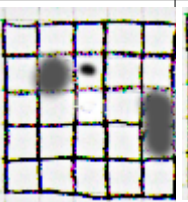
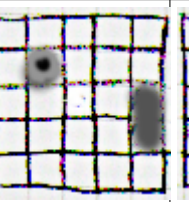
Интеллектуальные операции: ассоциации, сопоставление, сравнение — связь между разными текстами; синтез, кодирование, программирование — составление программ-текстов;

Шаг 4: Условные ветвления: строки

(программы в виде текстов). Ввести дополнительный объект: строки с командами могут быть помечены как выполняющиеся только когда текущая клетка [не]закрашена. Убедиться, что обучаемые могут выполнять/составлять такие программы.

	
Текст программы с условиями	То же, но в виде значков

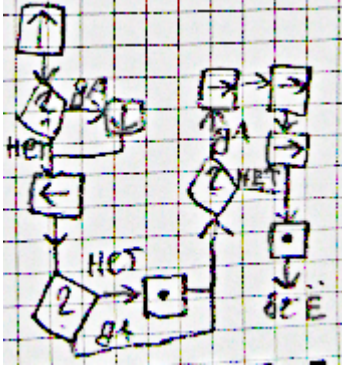
Шаги выполнения программы:

						
Начальное состояние	«Вверх»	«Закрашено ? Тогда вниз» - команда не выполняется	«Влево»	«Не закрашено? Тогда закрасить» - команда не выполняется	«Закрашено ? Тогда вправо вправо вправо» - выполняются все три «вправо».	«Закрасить»

Интеллектуальные операции: синтез, кодирование, программирование — составление программ-текстов; обобщение — выделение общего и частного в условных операторах.

Шаг 5: Условные ветвления: блок-схемы

(программы в виде диаграмм). Объяснение ветвления в диаграммах (ромбик вместо прямоугольника). Убедиться

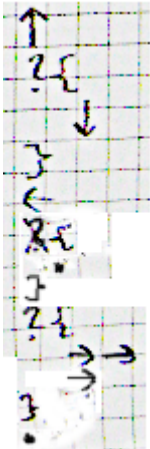


- Та же программа в виде блок-схемы.

Интеллектуальные операции: ассоциации, сопоставление, сравнение — связь программы с условными строками и программы в виде блок-схемы;

Шаг 6: Условные ветвления: снова текст.

В дополнение к тому нашему условному оператору, который действует на строку введём более сложный блочный оператор ветвления (со скобочками), который может охватывать много строк. Убедиться... Убедиться, что обучаемые понимают суть вложенных операторов ветвления (если не понимают, показать, что нужно считать скобки). Убедиться, что обучаемые понимают пустой оператор ветвления (который ничего не делает). Убедиться, что комбинирование нового оператора с тем, что был к шаге 4 проходит нормально.



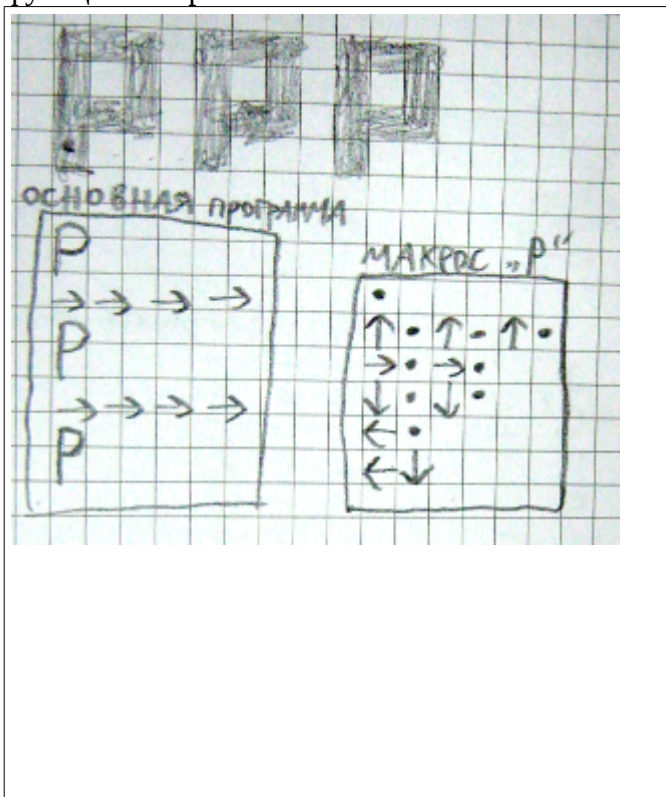

- Та же программа, но с «блочными» условиями («?» действует не на строку, а на блок в «{ }»)

Интеллектуальные операции: обобщение — выделение общего и частного в условных операторах в блочном виде; ассоциации, сопоставление, сравнение — связь программы с условными блоками с программами и двумя другими вышеприведёнными видами таких программ; различение — отличия блочного оператора от строчного;

Шаг 7: Макросы

(программы в виде текстов). Ввести дополнительную область (вдали от того места, где обычно записана программа), в которой записываются куски программ с именем. Показать как программа с макросами разворачивается в полную программу. Убедиться, что обучаемые могут разворачивать макросы, также выполнять программы, разворачивая их в уме.

Замечание: про процедуры объясняется во второй части, где про машинный код. Про функции - в третьей части.

	
<p>Рисунок в виде трёх букв «Р», которые робот должен нарисовать, программа, использующая макрос, который рисует одну букву «Р» и код этого макроса.</p>	<p>Развёрнутая форма этой программы — там, где было «Р» подставился код макроса.</p>

Интеллектуальные операции: синтез — составление полного текста программы из программы с макросами; группировка — выделение повторяющихся кусков в обычной программе и составление программы с макросами.

Шаг 8: Циклы: блок-схемы

Объяснить на диаграммах на основе "ромбика" и стрелки назад. Обязательно устроить цикл, который ни разу не выполняется, и вечный цикл. Проследить, чтобы при выполнении обучаемым зациквившейся программы прекращается только по явному указанию преподавателя (или присутствует комментарий про бесконечность такой программы).

/ Далее рисунки ещё не нарисованы */*

Интеллектуальные операции: (аналогично как в шаге 4)

Шаг 9: Циклы в тексте программ.

Добавить оператор цикла (пока клетка [не] закрашена) в текстовую часть программы. Убедиться ...

Интеллектуальные операции: (аналогично как в шаге 5)

Шаг 10: Циклы: счётчики.

Добавить конструкцию "Выполнять эту строку N раз". Убедиться. Явно обратить внимание,

что теперь нам нужно помнить "который сейчас раз".
Интеллектуальные операции: (аналогично как в шаге 4)

Шаг 11: Циклы со счётчиком: блок-схемы.

Те же программы, но в диаграммах. Теперь в ромбике может написано "Сейчас уже N-й раз?". Обратит внимание, что теперь из состояния поля и того, какая команда сейчас выполняется больше нельзя сделать однозначный вывод что будет происходить дальше (потому что ещё нужно знать который раз мы это выполняем). Это можно проиллюстрировать, давая другому обучаемому закончить выполнение прерванной программы (не видя до этого что происходило).

Интеллектуальные операции: (аналогично как в шаге 5)

Шаг 12: Переменные.

Рядом с полем, где робот (не около программы!) ввести окошко, к котором отображается то, который это раз мы выполняем инструкции в данном цикле (если вне цикла - в окошке пусто). Заметить теперь, что проблема с индетерминизмом (из шага 11) исчезла - мы можем посмотреть которая это итерация цикла в окошке. Несколько таких циклов - несколько окошек. Упомянуть, что состояние поля (положение робота, закрашенность клеток, числа в окошках) называется "состоянием". Показать, что состояния можно сохранять (перерисовывать всё поле плюс окошки) и восстанавливать (перерисовывать обратно) и что программа из одного состояния будет всегда выполняться одинаковым способом.

Интеллектуальные операции: сравнение, сопоставление — цикл из пункта 11 или 12 и цикл с использованием переменных.; понимание — понятие состояний программы.

Шаг 13: Именованные переменные.

1. Рядом с окошком написать имя переменной (придумать какое-либо, например, "который_раз").

2. В ромбике с нашим циклом написать вместо "Сейчас N-й раз?" написать "В окошке который_раз написано число меньше N?".

3. Добавить в диаграмму команду "Увеличим число в окошке который_раз".

Убедиться, что обучаемые правильно реагируют а. Если команду "Увеличим число" убрать (вечный цикл), б. Если из несколько (например, одна до нормальных команд, другая после, третья среди них).

Интеллектуальные операции: сравнение, сопоставление — тот же цикл, но более подробными словами.

Шаг 14: Циклы в тексте.

То же, но теперь мы может перенести циклы на текст (аналогично шагу 6).

Интеллектуальные операции: (аналогично как и в случае других перенесений на другую форму программ).

Шаг 15: Обработка ошибок.

Где-то около макросов ввести область с командами, которые будут выполняться если а. пытаемся закрасить клетку повторно, б. робот пытается выйти за границы поля (объявив эти ситуации аварийными).

Интеллектуальные операции: обобщение, интеллектуализация образа — ошибка теперь —

часть программы, которую мы обрабатываем примерно как и остальные команды.

Часть 2: Машинный код.

Эта часть должна показать обучаемым что происходит к компьютера внутри. Обучаемый должен понимать, что программы как в части 1 сначала переводятся в программы как в части 2, а потом уже выполняются, и особенности программ в части 2 (ограниченность ресурсов в первую очередь) сказывается и на «обычных» программах из 1-й части. Обучаемый должен понять что значит рекурсия, как такие программы выполняются, как программа может менять сама себя, почему память может переполниться и т.д.

В данной части используется немного другая модель: числа в клеточках. Система с клеточками одномерная (лента с клетками). Каждое число означает какую-то команду. По умолчанию везде нули.

Две клетки особые: 1. текущая команда (сдвигается вправо при выполнении команды), 2. курсор (аналог "робота" из первой части). Текущая команда пусть будет в левой части ленты (например, самая первая клетка), курсор - где-то посреди правой части ленты. Левая часть ленты - код, правая — данные.

Программы представлены теперь только числами в клеточках, блок-схем нету.

Пример системы команд:

- 1 - увеличить число в клетке под курсором на единицу.
- 2 - уменьшить число в клетке под курсором на единицу (ниже нуля или -1 или, например, 9).
- 3 - сдвинуть курсор вправо
- 4 - сдвинуть курсор влево
- 0 - конец программы

Шаг 1: простые программы.

Убедиться что простые программы выполняются и составляются успешно. Пример задания: "записать в левой части ленты такие числа, чтобы получилась программа, при выполнении которой в правой части оказались вот такие числа".

Пример простой программы и шагов её выполнения:

До начала выполнения программы	0 0 1 1 3 1 1 4 2 3 3 1 1 1 0 0 0 0 0 0 0 0 0 0
Выполнено 3 шага	0 0 1 1 3 1 1 4 2 3 3 1 1 1 0 0 0 0 0 0 2 0 0 0 0
Выполнено 7 шагов	0 0 1 1 3 1 1 4 2 3 3 1 1 1 0 0 0 0 0 0 1 2 0 0 0
После выполнения	0 0 1 1 3 1 1 4 2 3 3 1 1 1 0 0 0 0 0 0 1 2 3 0 0 0

программы	
-----------	--

Интеллектуальные операции: программирование; синтез, планирование, кодирование, принятие решения — как в шаге 2 первой части; предметное манипулирование — изменение циферок/передвижение стрелочек на доске согласно командам.

Шаг 2: больше команд.

- 5 - ничего не делать
- 6 - прыгнуть на 10 команд влево
- 7 - прыгнуть на 10 команд вправо
- 8 - прыгнуть на 10 команд влево, если число под курсором не 0
- 9 - прыгнуть на 10 команд вправо, если число под курсором не 0

Убедиться, что обучаемые могут придумывать более хитрые программы. Показать, что программа может изменить сама себя (так как она на той же ленте, где и код).

Интеллектуальные операции: понимание — единство программы и данных, которые она обрабатывает.

Шаг 3: адреса.

Пронумеровать все ячейки. Добавить команды, которые ссылаются на ячейки по их номеру
(to be continued)

Далее должны быть рассмотрены: стек и регистры, вызов процедур «изнутри». Стек, естественно, находится на той же самой ленте — появляется ещё третья «стрелочка». Потом снова вводим клеточки и робота, но теперь уже робот управляется не программой в виде текста или диаграмм, а командами с этой ленты. Должна рассматриваться рекурсия.

В третьей части предполагается рассмотрение функциональности: все объекты будут неизменяемые. Робот и клеточки также будут. Показать что все функции строят новое поле (с каким-то изменением), а старые стирать только через некоторое время с комментарием «они не нужны, потому что мы к ним больше не обращаемся и не обратимся». Понятие «состояния», введённого в первой части будет актуально. Можно рассмотреть списки, замыкания. Рекурсия теперь неотъемлемая часть.

Заключение

«Курс» чем дальше, тем сложнее. Некоторые вещи не рассмотрены (так как я сам пока их не изучил). Только первая часть даст общее представление о программировании на базовом уровне, достаточном для составления простых программ. Вторая часть добавит представление о том как оно это может происходить внутри (правильное понимание рекурсии, переполнение буфера, как работает вызов функции и т.д.). Третья часть создаст базу для последующего изучения более сложных и высокоуровневых языков (замыкания, функции как объект первого класса, и т. д.)

Ссылки

1. В.Д. Шадриков «Интеллектуальные операции»

http://new.hse.ru/sites/infospace/podrazd/facul/facul_psi/DocLib3/%D0%AD%D0%BB%D0%B5%D0%BA%D1%82%D1%80%D0%BE%D0%BD%D0%BD%D1%8B%D0%B5%20%D0%BF%D1%83%D0%B1%D0%BB%D0%B8%D0%BA%D0%B0%D1%86%D0%B8%D0%B8/%D0%A8%D0%B0%D0%B4%D1%80%D0%B8%D0%BA%D0%BE%D0%B2-%D0%98%D0%BD%D1%82%D0%B5%D0%BB%D0%BB%D0%B5%D0%BA%D1%82%D1%83%D0%B0%D0%BB%D1%8C%D0%BD%D1%8B%D0%B5%20%D0%BE%D0%BF%D0%B5%D1%80%D0%B0%D1%86%D0%B8%D0%B8.doc