УДК 519.6(075.8)

# ОБ АЛГОРИТМИЧЕСКОЙ ПОДГОТОВКЕ СТУДЕНТОВ



***Т.А. Сабаева, Д.Т. Чекмарев***

Нижегородский госуниверситет им. Н.И. Лобачевского

4ekm@mm.unn.ru

*Поступила в редакцию 17.11.2014*

Рассматривается проблема подготовки студентов в области разработки и реализации алгоритмов, а также связь данной проблемы с изменением системы вступительных экзаменов в вузы.

*Ключевые слова:* алгоритм, структуры данных, эффективность, методы построения, компьютер, самостоятельная работа.

Кардинальные изменения, произошедшие в средней школе за последние годы, привели к необходимости перестройки содержания некоторых предметов, преподаваемых на первых курсах. В работе рассматривается проблема алгоритмической подготовки студентов, включающая в себя знание порядка двух десятков базовых алгоритмов и способность самостоятельной их разработки для решения сложных задач. Анализируя имеющийся у авторов широкий опыт преподавания программирования и дисциплин, в которых оно используется (курс «Методы вычислений» и другие) на механико-математическом факультете ННГУ, можно сделать следующие выводы.

1. При существовавшей в 1990-е годы системе вступительных экзаменов на компьютерные и физико-математические специальности ННГУ данная проблема в значительной степени решалась за счет довузовской подготовки на подготовительных курсах ННГУ. Это связано со следующими факторами:

- наличием вступительного экзамена по информатике;
- сравнительно высоким конкурсом на компьютерные специальности;
- высоким уровнем и содержанием задач вступительных экзаменов [1].

2. В связи с отменой с 2008 года вступительного экзамена по информатике и переходом на ЕГЭ наблюдается существенное снижение способности использования базовых алгоритмов. Это связано прежде всего со слабой (и неодинаковой для разных школ) школьной подготовкой в данном направлении. Подтверждением этого служит существенная разница в усвоении курса программирования у студентов-математиков, сдававших и не сдававших информатику на вступительных экзаменах.

3. С целью сохранения (еще лучше – повышения) существующего уровня алгоритмической подготовки следует предусмотреть ряд мер как в самой высшей школе, так и во взаимодействии ее со средней школой. К ним можно отнести, например, следующее:

- переработку начального курса программирования с упором на алгоритмическую подготовку;
- организацию приема части студентов на компьютерные специальности через олимпиады по информатике и соответствующие подготовительные курсы к таким олимпиадам;
- используя опыт работы со школьниками, включающий в себя работу в приемной комиссии по информатике и на подготовительных курсах, по программе ННГУ «Одаренные дети» и др., организовать соответствующую подготовку в профильных классах средней школы.

Учебники и программы школ по информатике и информационным технологиям основной упор делают на максимальное изучение операционных систем (в основном Windows) и приложений, включенных в базовую комплектацию операционных систем. Это приводит к тому, что на первых курсах возникают проблемы при построении простейших алгоритмов, оценке их эффективности.

В практической деятельности при решении задач необходимы знания в той области, в которой формулируется данная задача, и знания в области математики для формализации поставленной задачи, то есть построения математической модели. Для этой модели разрабатывается алгоритм, проводится его оценка и далее разрабатывается его реализация на одном из языков программирования.

При проведении занятий перед преподавателями стоят следующие задачи:

- дать студентам навыки разработки алгоритмов на основе базовых, которые они освоили раньше;
- формировать способность к принятию самостоятельных решений, используя доступные им информационные ресурсы;
- дать навыки работы в группе.

Построение алгоритма можно разбить на следующие этапы, последовательное выполнение которых приведет к построению программы, максимально свободной от логических ошибок [2].

1. Постановка задачи с обсуждением терминологии, характерной для предметной области, и непротиворечивости описываемых явлений.
2. Построение математической модели и проверка ее адекватности поставленной задаче. Определение понятия решения построенной математической модели и понятия корректности решения.
3. Анализ существующих алгоритмов для решения подобных задач. Оценка трудностей, не позволяющих использовать эти алгоритмы при решении поставленной задачи.
4. Разработка собственного алгоритма для решения поставленной задачи и оценка его эффективности. Обязательно отметить, что построение алгоритма является задачей, не имеющей единственного решения, что приводит к обязательной оценке эффективности различных алгоритмов. Анализ эквивалентности решения, полученного построенным алгоритмом и решения соответствующей модели.
5. Доказательство правильности алгоритма. В этом пункте необходимо показать, что разработанный алгоритм дает искомый результат в пределах допущений, сделанных при постановке задачи.
6. Способы описания алгоритма (базовые структуры, метод разработки сверху вниз, иерархические структуры и так далее).
7. Реализация алгоритма (построение программы) и требования к технической документации, сопровождающей программный продукт. Разработка программы отладки и тестирования программного продукта.
8. Проведение отладки и тестирования.
9. Документация и презентация программного продукта.

При разработке алгоритмов существует несколько подходов. Три основных метода – это метод частных целей, метод подъема и метод отработки назад. В изолированном виде эти методы встречаются редко, так как чаще всего при разработке алгоритмов используется их комбинация [2, 3, 4]. Следует отметить, что в работе [2] активно используются игровые примеры, что также позволяет возбудить интерес у недавних школьников. Например, метод частных целей связан с разбиением исходной задачи на последовательное решение более простых задач – в надежде, что они требуют менее сложного решения. Однако выбор этих задач скорее вопрос интуиции и искусства и трудно алгоритмизируется. Частные задачи могут быть установлены при ответе на следующие вопросы:

– можно ли в задаче выделить инварианты;
– можно ли решить задачу для частных случаев;
– можно ли упростить задачу, наложив дополнительные ограничения;
– существуют ли решения подобных задач и нельзя ли эти решения модифицировать при решении поставленной задачи.

В качестве иллюстрации метода частных целей рассмотрим игровую задачу о преобразовании шведского креста в прямоугольник [2]. Задан крест, состоящий из пяти квадратов. Двумя линиями его следует разделить на фигуры, из которых можно составить прямоугольник, высота которого вдвое больше чем основание (рис.).

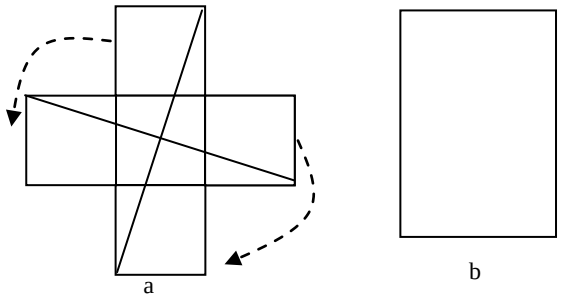


Рис.: a – исходный крест, b – искомый прямоугольник

Прежде всего рассмотрим, какие инварианты присутствуют в данной задаче. Ответ практически очевиден – это площадь фигур. Это утверждение приводит ко второй задаче: какое соотношение существует между стороной квадратов x, из которых состоит шведский крест, и меньшей стороной прямоугольника $y$. Площадь креста $5x^2$, а площадь прямоугольника $2y^2$. Таким образом, меньшая сторона прямоугольника определяется соотношением $y = \sqrt{10}x/2$. Следующий вопрос: где на фигуре рис. «а» находится отрезок длиной $y$ или $2y$. Легко видеть, что это линии, показанные на этом рисунке. Далее, необходимо показать, что эти линии перпендикулярны и делятся пополам (соединяя концы линий, получим квадрат), разрезать фигуру и переместить ее часть

так, как показано стрелками. Основанием полученного прямоугольника будет половина данного отрезка, то есть *y*.

Метод отработки назад хорошо знаком школьникам, решающим трудные задачи. Посмотреть в ответ и пытаться от него перейти к условиям задачи. Если обратное восстановление возможно, то это будет алгоритмом решения.

При реализации алгоритмов рассматриваются возможные структуры данных и численные методы и оценка их эффективности. В качестве базового языка используется C++. В начальных курсах программирования наибольшее внимание уделяется таким темам, как принципы работы компьютера, синтаксис языка и так далее. Однако есть некоторые разделы, на которые необходимо обратить внимание, часть материала которых можно перенести на самостоятельную работу или на семинарские занятия. В частности, при проведении практических занятий формирование группы из нескольких человек для решения объемной задачи позволяет вырабатывать навыки командной работы. Совместная работа позволяет выработать принципы общения и обсуждения возникающих проблем, принятия коллективного решения.

Для поддержания их работы базовые алгоритмы и их реализация изложены в учебном пособии [5], использование которого в учебном процессе дает студентам возможность конструирования собственных алгоритмов, так как в нем представлены наборы задач и рекомендации по их оформлению. Для проведения оценки разработанного алгоритма студентам предлагается использовать методологию, приведенную в работах [2, 3, 4].

*Список литературы*

1. Барышева И.В. и др. Информатика для абитуриента. Задачи и решения // И.В. Барышева, С.Ю. Гергель, С.Ю. Городецкий, В.А. Гришагин, В.С. Громницкий, Г.А. Долгов, А.П. Кулакова, В.И. Малыженков, М.В. Маркина, В.Л. Тарасов, И.А. Фомина. Нижний Новгород: Нижегородский госуниверситет, 2007. 360 с.

2. Гудман С., Хидетниеми С. Введение в разработку и анализ алгоритмов // М.: Мир, 1981. 368 с.

3. Кормен Т. и др. Алгоритмы. Построение и анализ // Т. Кормен, Ч. Лейзерсон, Р. Ривест, К. Штайн. М.: Вильямс, 2005. 1269 с.

4. Кушниренко А.Г., Лебедев Г.В. Программирование для математиков // М.: Наука, 1988. 382 с.

5. Перова В.И., Сабаева Т.А. Программирование на языке C++ // Нижний Новгород: Изд-во Нижегородского госуниверситета, 2013. 132 с.

## ON THE ALGORITHMIC TRAINING OF STUDENTS

***T.A. Sabaeva, D.T. Chekmaryov***

The problem of training of students in the development and implementation of algorithms is considered. We examine this problem in relation to the changes in the system of entrance examinations to universities.

*Keywords:* algorithm, data structure, efficiency, methods of construction, computer, independent work.

*References*

1. Barysheva I.V. i dr. Informatika dlia abiturienta. Zadachi i resheniia // I.V. Barysheva, S.Iu. Gergel', S.Iu. Gorodetskii, V.A. Grishagin, V.S. Gromnitskii, G.A. Dolgov, A.P. Kulakova, V.I. Malyzhenkov, M.V. Markina, V.L. Tarasov, I.A. Fomina. Nizhnii Novgorod: Nizhegorodskii gosuniversitet, 2007. 360 s.

2. Gudman S., Khidetniemi S. Vvedenie v razrabotku i analiz algoritmov // M.: Mir, 1981. 368 s.

3. Kormen T. i dr. Algoritmy. Postroenie i analiz // T. Kormen, Ch. Leizerson, R. Rivest, K. Shtain. M.: Vil'iams, 2005. 1269 s.

4. Kushnirenko A.G., Lebedev G.V. Programmirovanie dlia matematikov // M.: Nauka, 1988. 382 s.

5. Perova V.I., Sabaeva T.A. Programmirovanie na iazyke S++ // Nizhnii Novgorod: Izd-vo Nizhegorodskogo gosuniversiteta, 2013. 132 s.