УДК 377.6

ОСОБЕННОСТИ ВЫБОРА ЗАДАЧ ПРИ ИЗУЧЕНИИ ОСНОВ ПРОГРАММИРОВАНИЯ В СПО

Беспалько А.А., Сочнева Н.В., Тощев А.П.

Волго-Вятский филиал Московского института связи и информатики, Нижний Новгород, e-mail: tuola@list.ru

В статье рассматриваются проблемы и пути решения обучения основам программирования. Большое внимание уделено особенностям выбора задач для обучения программированию на первом этапе обучения студентов среднего профессионального образования. Предлагаются примеры и подробно разбираются виды заданий, рассматриваются формируемые компетенции. Обосновывается выбор языка программирования, приводятся примеры компиляторов и сред исполнения программ. Важно обеспечить уровень обучения алгоритмике на таком уровне, чтобы учащиеся смогли перестроить мышление и начать думать «алгоритмически». Каждая задача в реальной жизни требует тщательного продумывания, обучение программированию способствует этому. Подбор видов заданий имеет решающее значение для достижения этой цели. В статье предлагаются три типа: математические (простые, используются на начальном этапе каждой темы), задачи с неочевидным условием (с нетривиальным условием, но основанные на типовых алгоритмах), стандартные алгоритмы. Итогом курса является разработка проекта – большой программы, в которой консолидируются все усвоенные за семестр знания. Поэтапный плавный подход снижает потерю интереса в течение семестра, повышает качество образовательного процесса и дает возможность организовать постепенное погружение в темы.

Ключевые слова: программирование, язык программирования, алгоритм, модель, блок-схемы

FEATURES OF THE CHOICE OF TASKS WHEN STUDYING THE BASICS OF PROGRAMMING IN SECONDARY VOCATIONAL EDUCATION

Bespalko A.A., Sochneva N.V., Toschev A.P.

Volga-Vyatka branch of the Moscow Institute of communications and Informatics, Nizhniy Novgorod, e-mail: tuola@list.ru

The article deals with the problems and solutions of teaching the basics of programming. Much attention is paid to the peculiarities of choosing tasks for teaching programming at the first stage of training students of secondary vocational education. Examples are offered and the types of tasks are analyzed in detail, and the competencies that are being formed are considered. The choice of programming language is justified, and examples of compilers and runtime environments are given. It is important to ensure that the level of training in Algorithmics is at such a level that students can rebuild their thinking and start thinking «algorithmically». Every task in real life requires careful thinking, and training in programming contributes to this. Selecting the types of tasks is crucial to achieving this goal. The article offers three types: mathematical (simple, used at the initial stage of each topic), problems with a non-obvious condition (with a non-trivial condition, but based on standard algorithms), and standard algorithms. The result of the course is the development of a project-a large program that consolidates all the knowledge acquired during the semester. A step-by-step smooth approach reduces the loss of interest during the semester, improves the quality of the educational process, and makes it possible to organize a gradual immersion in topics.

Keywords: programming, programming language, algorithm, model, flowcharts

Язык программирования – это искусственно созданный инструмент, который необходим, чтобы представить готовый алгоритм для некоего исполнителя. Однако искусство программирования включает в себя не только написание инструкций исходя из семантических и синтаксических правил языка программирования, но и построение математической или структурной модели задачи, выбор метода ее реализации и только затем написание оптимального кода.

Проблема обучения учащихся среднепрофессионального образования стоит очень остро, особенно если речь идет о непрофильных специальностях. Часто студентам бывает сложно воспринять новое мышление, и при решении даже самых простых задач возникают сложности. В таких случаях требуется тщательно проработанный поэтапный подход.

Материалы и методы исследования

На первом этапе обучения программированию учащиеся осваивают предмет «Основы алгоритмизации и программирования». Студенты знакомятся с предметом и учатся строить базовые алгоритмические структуры: последовательный алгоритм, ветвление, цикл, работа с массивами. Как правило, для освоения подобного материала используются простые задачи, чаще математические. Пройдя большой путь в обучении веб-технологиям, мы и здесь выбрали практико-ориентированный подход, поставив в приоритет умение моделировать задачу [1].

Шаги построения алгоритмов

Стандартная последовательность решения поставленной задачи включает в себя ряд шагов [2].

1. Постановка задачи, разработка математической модели.
2. Выбор метода численного решения.
3. Построение алгоритма.
4. Разработка программы.
5. Отладка и испытание программы.
6. Решение задачи на ЭВМ, обработка и оформление результатов расчета.

Первый шаг является определяющим для эффективного прохождения последующих этапов. Поэтому важно развить у студентов алгоритмическое мышление.

Алгоритмическое мышление – это умение мыслить последовательно, выстраивая полную схему решения задачи и разбивая ее на отдельные структурные элементы [3]. Это именно свойство мышления, которое прививается в самом начале обучения. Без него получение компетенций программиста невозможно.

В ходе преподавания было выделено три класса задач, которые предлагаются на первом этапе обучения.

Математические

Представляют собой построение алгоритмов для решения простейших математических задач из курса школьной алгебры и геометрии, что упрощает построение математической модели. Учащиеся прекрасно знакомы с основными формулами: нахождение площади, периметра, объема, угла и т.д. В этот момент важно показать осмысленное прохождение шагов решения задачи, которые обычно студент делает неосознанно.

Построение модели решения задачи совпадает с этапом выбора численных методов решения. Достаточно найти формулу – и решение уже перед глазами. В этот момент важно показать, что такое входные и выходные данные. Рассмотрим на примере.

Найдите площадь параллелограмма, если одна его сторона равна 15 см, вторая 24 см, а угол между ними составляет 30 градусов.

Решение этой задачи элементарно. Воспользуемся формулой $S = a*b*\sin(x)$ и получим ответ $S = 180$. В уме студент решает эту задачу за долю секунды.

При переходе к построению алгоритма следует напомнить важное свойство, которым должен обладать алгоритм: массовость. Не имеет смысла писать программу, которая вычисляет площадь именно этого конкретного параллелограмма с заданными параметрами. Чтобы алгоритм был массовым, он должен решить целый класс подобных задач. Это значит, что в программировании условие будет звучать следующим образом: найдите площадь параллелограмма по двум сторонам и углу между ними.

Простая расчетная задача в первый момент ставит в тупик тех, кто еще не начал программировать: откуда мы знаем исходные значения. Важный этап для педагога – расписать и подробно объяснить ход мысли при алгоритмическом мышлении.

1. Определить математическую модель задачи.
2. Найти численные методы решения.
3. Перечислить исходные и выходные данные.
4. Выяснить, что нужно знать для достижения цели.

Последние два шага для начинающих программистов не всегда являются очевидными. Где брать и как обозначать исходные значения? Их должен дать тот, кто заинтересован в решении задачи – пользователь.

На таком простом примере преподаватель объясняет ход мысли, оптимальную последовательность решения. И при кажущейся простоте привить этот образ мышления бывает достаточно сложно – это достигается постоянным проговариванием и оформлением решения каждой задачи. Даже успевающие ученики, которые опережают группу, часто проходят эти шаги поверхностно, не осмысляя их.

По результатам решения строится блок-схема. Этот этап важен для визуализации последовательного решения. Алгоритм всегда состоит из конечного числа шагов, и именно на примере блок-схем студенты учатся понимать это.

Еще один важный момент – введение понятий переменной и операции присвоения. При создании программы это базовые знания, поэтому понимать механизм действий с памятью необходимо. Решая задачу, учащиеся знакомятся с этими понятиями и учатся ими оперировать. Элементарные понятия: именование переменной, порядок присваивания.

Задачи с неочевидным условием

К таким задачам мы относим большинство задач без очевидных математических формул. Модель приходится строить самостоятельно. От качества построенной модели зависит оптимальность построения алгоритма, а значит, решение задачи. Для получения навыков такой алгоритмизации мы используем задачи на жизненный опыт. В качестве базового задачника мы выбрали сборник задач [4].

Большинство заданий позволяет показать, как программирование применяется

в разных сферах человеческой деятельности. Большим плюсом является простота решения при кажущейся сложности формулировки.

В качестве примера выберем задачу с экономическим уклоном на построение последовательного алгоритма.

На изготовление платья идет 3 метра ткани, 1 катушка ниток, 8 пуговиц и застежка-молния. Найти прибыль от продажи платья.

Для поиска решения вместе со студентами преподаватель озвучивает последовательность шагов.

1. Что такое прибыль? Это разница между ценой, за которую продали платье и его себестоимостью.

2. Чтобы найти прибыль, необходимо знать цену продажи и себестоимость.

3. Цену продажи определяет тот, кто продает, – пользователь.

4. Себестоимость складывается из цены и количества израсходованных материалов.

5. Необходимо знать цену метра ткани, одной пуговицы, молнии и катушки ниток. Эти данные тоже задает пользователь.

Таким образом, алгоритм строится на пяти исходных значений. После того, как они обозначены конкретными именами, легко построить математическую модель решения:

$$Prib = PricePl - (3*PrT + 8PrP + PrN + PrM),$$

где *PricePl* – цена продажи платья,
PrT – стоимость метра ткани,
PrN – стоимость катушки ниток,
PrM – стоимость молнии.

На примере этой задачи студенты учатся называть переменные осмысленно, ведь при таком количестве входных параметров легко запутаться. Например, обозначив переменные как *a*, *b*, *c*, *d*, *p* учащийся быстро забудет, какие именно данные хранит в себе переменная. Кажущаяся скорость написания приводит к потере времени в процессе отладки и тестировании программы. А когда речь идет о задачах, на решение которых требуется не один день, фактор именования переменных имеет огромное значение.

При решении задач программирования важно понимание практической пользы. «Нарешивание» десятков ничего не значащих задачек обычно вызывает скуку и ощущение рутины. Поэтому в нашем курсе часть времени отводится на осмысление практической значимости.

Студенты отвечают на вопрос и ищут ситуации, когда эта программа становится полезной. Например, швея, которая шьет платья на продажу, может повысить прибыль, сэкономив на затратах и выбрав более дешевый материал. Вводя разные параметры, она может рассчитать самый выгодный вариант и подобрать оптимальную цену продажи платья.

Закономерным предложением является отказ от фиксированного количества расходных материалов и добавление этого параметра в набор исходных данных, которые задает пользователь. Модель увеличивается, но при этом алгоритм становится универсальным.

Стандартные алгоритмы

Многие задачи строятся на поиске суммы, количества, минимума и максимума, сортировки. Это стандартные алгоритмы, которые являются типовыми для любых задач [5] – математических и практических. По аналогии, на первом этапе пишутся алгоритмы математического характера – это помогает быстро понять механизм поиска и подсчета. Например, найти количество корней функции на заданном промежутке, определить минимальное и максимальное значение функции.

Далее вводятся программы практического значения. Это могут быть игры с пользователем, к примеру «Угадай число»: игроку дается 10 попыток, после чего выводятся количество неправильных ответов, с какой попытки угадано задуманное.

Можно смоделировать сдачу теста с подсчетом итогового результата и выставления оценки.

Одним из самых распространенных вариантов освоения массивов являются задачи на матрицы. Математический аппарат предлагает множество подзадач, но у них есть один большой недостаток – ограниченность. Решив одну задачу с точки зрения массовости и универсальности, учащиеся закрывают тему. А этого недостаточно для глубокого освоения алгоритмических структур. Поэтому мы активно используем практические задачи на массивы.

В качестве примера приведем две задачи: на одномерный и двумерный массивы.

1. Школьник ежедневно покупает в типографии газеты и затем продает их с небольшой наценкой. Когда газеты покупают хорошо, он имеет неплохую прибыль, но иногда торговля идет плохо, так что даже не окупаются затраты на покупку газет в типографии. Зная ежедневную прибыль (которая не всегда идет со знаком «+»), найти прибыль за месяц. Ежедневную сумму прибыли организовать с помощью массива.

2. Владелец пяти хлебобулочных киосков в разных районах города держит в них одинаковый ассортимент товаров из 20 наименований. Утром в каждый киоск завозят

по 100 единиц каждой продукции. Составить таблицу наличия ассортимента по всем киоскам в середине дня. В качестве первого индекса взять номер киоска, а вторым – номер товара по ассортименту. По таблице наличия ассортимента хлебобулочных изделий определить:

– Какой вид продукции следует закупать на хлебокомбинате в первую очередь?

– Какой вид продукции пользуется наименьшим спросом у покупателей?

– В какой киоск и какой вид продукции следует завезти в первую очередь?

– В каком киоске лучше всего идет торговля?

– В каком киоске хуже всего идет торговля?

– Составить новую таблицу, содержащую данные о том, в какой киоск и сколько следует завезти того или иного товара для заполнения ассортимента до нормы (каждого вида продукции в каждом киоске должно быть 100 единиц).

Итоговая задача по курсу

В качестве подведения итога студентам предлагается самостоятельная разработка уже целого проекта, в котором применяются все изученные ранее структуры и подходы. Она имеет практическую значимость и интересна для учащихся, строится на проектном подходе [6].

В процессе формулировки задания мы внимательно оцениваем его полноту. Важно соблюсти равновесие: не предоставить слишком подробное описание, но раскрыть все необходимые аспекты.

Одна из любимых задач студентов после 18 лет – определение размера штрафа с радара измерения скорости. Приведем условие.

На участке дороги установлен радар скорости, который настраивается в зависимости от текущего ограничения. Например, если на этом участке проводятся дорожные работы, то оно может составлять 40 км/ч, в обычном состоянии дороги – 90 км/ч. Построить таблицу санкций для 100 водителей, которые попали под действие радара. Машина идентифицируется по номеру. Условия для начисления штрафов:

Превышение до 30 км/ч – предупреждение.

Превышение от 30 до 40 км/ч – штраф 500 рублей.

Превышение от 40 до 50 км/ч – штраф 1000 рублей.

Превышение от 50 до 60 км/ч – штраф 2000 рублей.

Превышение свыше 60 км/ч – лишение прав.

Вывести на экран таблицу, включающую номер машины, скорость, превышение, сумму штрафа, комментарий.

Плюсом этой задачи является ее вариативность. Она дает большой простор для самостоятельной работы студентов.

1. Найти условия для наложения взысканий, используя ПДД.

2. Распределить данные по массивам, определяясь с их структурой и количеством.

3. Выбрать способ заполнения таблицы – случайным образом или с клавиатуры (оптимально – случайные числа, так как объем данных велик).

4. Вводить ли проверку на проезд одной и той же машины несколько раз и стоит ли ограничивать это число.

Если в курсе изучались регулярные выражения, то можно организовать с их помощью проверку корректного ввода данных. Задачу можно упростить или усложнить по необходимости.

Результаты исследования и их обсуждение

Мы считаем такой подход к преподаванию основ алгоритмизации оптимальным. Он учит студентов широко мыслить и находить алгоритмы для любых поставленных задач. Постепенное поэтапное погружение в мир программирования мягко формирует алгоритмическое мышление, не вызывая отторжения процесса. Тексты задач зачастую похожи на головоломку, их хочется понять и разгадать. Наш опыт показал, что студентам намного интереснее решать не абстрактные примеры, а брать алгоритмы «из жизни». Итогом такого подхода становится развитие важных компетенций в сфере алгоритмики и программирования.

При этом выбор языка программирования не имеет значения. Базовые структуры обладают одним и тем же функционалом в любом из них. Авторы на занятиях предпочитают C, так как он лежит в основе большинства распространенных языков программирования, его синтаксис является исходным, написано множество вспомогательной литературы [7]. Кроме того, в свободном доступе, в том числе онлайн, имеется много компиляторов и сред исполнения, а также IDE, например CodeBlock, Dev C++ и т.д. Найти исполнитель можно даже для мобильных гаджетов, что существенно расширяет возможности для дистанционного обучения. Стоит заметить, что создано множество пособий и вспомогательных материалов и по Паскалю [8].

Заключение

Итогом введения множества задач с практическим уклоном в освоение азов

программирования является гибкость мышления при формировании алгоритмики. Любая поставленная задача рассматривается как совокупность данных и моделей, которые можно использовать для построения алгоритма.

Список литературы

1. Беспалько А.А., Сочнева Н.В. Проблема выбора методов обучения разработке WEB-приложений в системе СПО // European Social Science Journal. 2018. № 5–1. С. 250–254.

2. Сундукова Т.О., Ваныкина Г.В. Структуры и алгоритмы компьютерной обработки данных: учебное пособие. 3-е изд. Саратов: Ай Пи Ар Медиа, 2020. 804 с.

3. Канатьева Е.С. Понятие алгоритмического мышления // Научное сообщество студентов: междисциплинарные исследования. Сборник статей по материалам XXII междунар. студ. науч.-практ. конф. № 11(22). [Электронный ресурс]. URL: https://sibac.info/archive/meghdis/11(22).pdf (дата обращения: 15.05.2020).

4. Беспалько И.И., Беспалько А.А. Зауральский институт информатики. Шадринск, 1999. 184 с.

5. Вирт Никлаус. Алгоритмы и структуры данных / Перевод Ф.В. Ткачева. 2-е изд. Саратов: Профобразование, 2019. 272 с.

6. Проектное обучение: практики внедрения в университетах / Ред. О.В. Лешуков, Н.В. Исаева, Л.А. Евстратова. М.: ИД Высшей школы экономики, 2018. 150 с.

7. Зоткин С.П. Программирование на языке высокого уровня C/C++: конспект лекций. 3-е изд. М.: МИСИ – МГСУ, ЭБС АСВ, 2018. 140 с.

8. Тюльпинова Н.В. Алгоритмизация и программирование: учебное пособие. Саратов: Вузовское образование, 2019. 200 с.