УДК 519.713

Ю.С. МАРЧЕНКО

ПРОГРАММИРОВАНИЕ. ПРОБЛЕМЫ В ОБУЧЕНИИ И МЕТОДИЧЕСКИЕ ОШИБКИ. СООБЩЕНИЕ 1

Описывается многолетний опыт преподавания различных курсов программирования. Анализируются проблемы, возникающие при обучении, обсуждаются возможные методические ошибки, которые допускают обучающие этой дисциплине.

1. Введение

Курсы информатики и основ программирования сегодня в программах общеобразовательных школ, средних специальных и высших учебных заведений практически всех специальностей. Основные навыки работы на компьютере, знание основ программирования необходимы сегодня врачу и строителю, биологу и механику, т.е. специалистам самых разных специальностей, когда необходимо выполнить обработку статистических данных, инженерные расчеты строительных конструкций, механических узлов, электронных схем.

2. Анализ литературных источников

Проанализировав лучшие, наиболее востребованные книги по программированию, издающиеся огромными тиражами, приходишь к выводу, что в них отсутствуют темы, дающие ключ к пониманию тех глав, с которых фактически начинаются эти книги.

Так, Г. Шилдт [1] сразу переходит к рассмотрению основных элементов языка, типов данных, инструкций управления, работе с массивами. Далее рассматриваются темы «Указатели» и «Функции».

Книга С. Прата [2], прежде всего, предлагает освоить базовую структуру программы на С++, и только после этого перейти к изучению деталей, например, циклов, объектов. В этой же главе рассматриваются функции и классы, которым в последующих главах будет уделено особое внимание. Далее автор переходит к рассмотрению встроенных типов данных и указывает на возможность языка С++ разрабатывать собственные типы данных.

В книге Р. Лафоре [3] изложение ведется в следующей последовательности. Вначале рассмотрены основы процедурного программирования, а затем – непосредственно изучение классов. Рассмотрению классов предшествует рассмотрение структур. Причиной этого является то, что класс представляет собой синтаксическое расширение структуры. Заметим, язык С++ предоставляет программисту возможность работать не только с данными, но и с адресами этих данных. Программа на языке С++ с использованием указателей более эффективна, чем вариант этой же программы без использования указателей. Сравнительная эффективность программ оценивается по времени выполнения и требуемой для программы памяти. Однако указатели не являются необходимыми при изучении основ объектно-ориентированного программирования (ООП) и, как правило, представляют трудность для понимания. Поэтому в [3] вполне обоснованно указатели рассматриваются позже, чем в большинстве книг по программированию на С++, пока не будут рассмотрены основные концепции ООП.

В книге авторов Х.М. Дейтел и П.Дж. Дейтел [4] вначале рассмотрены несколько простых программ с подробным их комментарием. Такое начало в изложении курса программирования является хорошим методическим приемом, дающим обучающему возможность увидеть и получить общее впечатление от программы на языке С++. Авторы называют это подходом «живого кода» [4, с.41]. Далее они очень кратко определяют понятие алгоритма [4, с. 152] и упоминают [4, с. 154] работу Бома и Джакопини [5], согласно которой алгоритм любой сложности может быть реализован с использованием трех типовых алгоритмических структур – последовательной, выбора и повторения. После этого авторы переходят к рассмотрению основных операторов языка.

Книга У. Савитча [6] представляет собой учебный курс по программированию на С++. Ее отличает взвешенный методический подход, который позволяет использовать классы с первых шагов в программировании. Облегчает изучение С++ большое количество примеров, учебных программных проектов, подсказки и предостережения. Автор отмечает, что разработчики больших систем программного обеспечения (например, операционных систем, компиляторов), прикладные программисты подразделяют процесс разработки на ряд этапов [6, с. 34], которые называют жизненным циклом программного обеспечения. У. Савитч описывает ряд общих принципов [6, с. 31-34], которые можно использовать при разработке программ независимо от языка программирования. Но, обратите внимание на «скромные» 4 страницы в книге объемом 704 страницы, которые отведены этой важной теме!

Общим недостатком этих, бесспорно лучших, книг по программированию на языке С++ является невосприятие многими обучающимися такой последовательности в изложении. Рассмотрению основных конструкций языка должны предшествовать темы, к обсуждению которых мы и переходим.

Таким образом, целью данной работы является обоснование последовательности изложения лекционного материала и практических занятий, при которой достигается оптимальное восприятие дисциплины «Основы программирования».

Прежде всего, начиная изучать программирование, обучаемый должен знать определение алгоритма и его свойства, понимать этапы, по которым следует провести решаемую на компьютере задачу. Ведь авторы указанных книг уверяют, что для освоения излагаемого материала не требуется предварительного опыта в программировании.

3. Алгоритм. Основные свойства

Предварительно алгоритм можно определить как последовательность некоторых правил (действий), приводящих к получению результата. Так, инструкция пользования телефо-

ном-автоматом, рецептура приготовления лекарств, кулинарные рецепты – это алгоритмы, которые окружают нас. При решении различных задач каждый алгоритм предполагает наличие некоторых исходных данных и приводит к получению искомого результата. Каждый алгоритм предписывает некоторую конечную цепочку действий (шагов). Такая цепочка называется алгоритмическим процессом.

Рассмотрим основные свойства алгоритмов. Как правило, алгоритм должен быть допустим для различных вариантов исходных данных (допустимые исходные данные). Это свойство называется массовостью. Так, вычисляя сумму элементов массива, тип массива может быть, например, целым, вещественным, логическим; количество элементов массива – произвольным конечным числом.

Под детерминированностью (определенностью) алгоритма понимают однозначность определения каждого шага (действия). Каждый шаг алгоритма должен быть сформулирован так, чтобы не допускать двузначности толкования. Так, высказывание «Мать любит дочь» может быть истолковано двояко. Такое высказывание недопустимо при определении алгоритма, так как его реализация с таким шагом может привести к различным результатам.

В начале XX века алгоритмы стали объектом изучения. Результаты, полученные в теории алгоритмов, служат фундаментом компьютерной технологии. Мы же кратко рассмотрели алгоритмы в их прикладном понимании.

4. Основные этапы решения задач на ЭВМ

В настоящее время на ЭВМ решают самые разнообразные задачи:
– расчет баллистических траекторий космических объектов;
– разработка компьютерных игр;
– обработка статистических данных;
– сложные инженерные расчеты в строительстве, механике, электронике;
– автоматизированное проектирование и многие другие задачи.

Но, несмотря на огромное разнообразие этих задач, усматривается определенная общность в процессе разработки, и можно выделить последовательность этапов, через которые нужно провести решаемую на ЭВМ задачу.

5. Постановка задачи

Под постановкой задачи понимают математическую или иную строгую формулировку решаемой задачи. Этот этап включает определение целей создаваемой программы, а также ограничений, предъявляемых к программе. При постановке задачи должны быть определены требования:
– ко времени решения поставленной задачи;
– к объему необходимых ресурсов, например, оперативной памяти (ОП);
– к необходимой точности вычислений.

Так, часто не принципиально за 10 или за 15 секунд получено решение задачи. Если же ЭВМ работает в режиме реального времени – управляет технологическим процессом (химическое производство, ядерные исследования), под управлением ЭВМ осуществляется запуск космического корабля, то на время ответа вычислительной системы могут накладываться жесткие временные ограничения, и это должно быть учтено при проектировании программы.

Если, например, объем ОП недостаточен для размещения всей программы, она размещается во внешней памяти и отдельными фрагментами загружается в ОП. Языковые средства C++ позволяют по-разному строить такую программу. Так, последующий фрагмент может замещать (перекрывать) предыдущий или располагаться в ОП вслед за предыдущим фрагментом. Таким образом, структура разрабатываемой программы определяется на этапе постановки задачи с возможной коррекцией на последующих этапах.

Вычисления, проводимые программой, например, на C++, могут быть выполнены с обычной, двойной точностью и особо точные. Это обеспечивает соответственно точность вычислений до 7, 15 и 19 значащих десятичных цифр. Поэтому точность вычислений должна быть обсуждена при постановке задачи.

6. Проектирование программы

На этом этапе следует выбрать метод расчета, если задача вычислительного характера. Если разрабатывается компьютерная игра, то должен быть определен ее сценарий. В любом случае следует выбрать или создать формальную модель, которая реализуется в разрабатываемой программе. На этапе проектирования определяют базовые и производные типы данных, с которыми будет работать программа. Наконец, определяют основные модули, из которых будет состоять программа и характер связей между этими модулями.

Рассмотрим пример вычисления sin x. Запишем разложение этой функции в ряд:

$$\sin x \approx x - \frac{x^3}{3!} + \frac{x^5}{5!} - \ldots + \frac{(-1)^{n+1} x^{n+1}}{(2n-1)!}.$$

В данном случае выбор численного метода заключается в том, что вычисление sin x сведено к вычислению алгебраической суммы членов ряда. Формирование суммы будем продолжать до тех пор, пока очередное слагаемое по абсолютной величине станет меньше заданной точности вычисления. Заметим, что при любом x, начиная с некоторого слагаемого, они будут монотонно убывать по абсолютной величине. Таким образом, вычисление тригонометрической функции сведено к вычислению алгебраической суммы членов ряда, где используются операции, которые выполняет ЭВМ.

7. Разработка алгоритма

Алгоритм – ключевое понятие в программировании. На этом этапе следует разработать детали проекта программы, когда программирование отдельных ее частей не вызывает никаких затруднений. Таким образом, алгоритм определяют как точное предписание, определяющее процесс преобразования за конечное число шагов исходных данных в искомый результат.

Существуют различные формы записи алгоритмов – словесно-формульная, графическая.

8. Кодирование алгоритма

После разработки алгоритма его записывают на языке, понятном ЭВМ. Это могут быть:

– язык символического представления реального машинного кода – Ассемблер;

– алгоритмические языки среднего и высокого уровней.

Разработка программы на языке Ассемблер требует знания логической организации (архитектуры) ЭВМ – адресность, регистры, флаги. Но программа, написанная на машинном языке, более эффективна, чем вариант этой программы на языке высокого уровня – она требует меньше памяти и быстрее выполняется.

Алгоритмические языки высокого уровня (например, Pascal) наиболее «дружелюбны» к программисту и выглядят как обычный текст на английском языке с использованием общепринятых математических знаков.

Языки среднего уровня (C, C++) сочетают «дружелюбность» языков высокого уровня с возможностью работы с отдельными битами, байтами, адресами, т.е. базовыми элементами ЭВМ, что характерно при программировании на Ассемблере. Код программы на C++ по времени выполнения и требуемой памяти сравним с программами, написанными на Ассемблере. Заметим, что язык C++ предусматривает в коде возможность применения ассемблерных вставок.

Использование алгоритмических языков предполагает наличие в компьютере соответствующего программного обеспечения (ПО). Так, преобразование программ с языка ассемблер в машинный код выполняет системная программа транслятор. Системные программы для преобразования последовательности операторов на языке высокого или среднего уровня в машинный код называются компиляторами.

ПО ЭВМ также должно включать компоновщики (редакторы связей), программные библиотеки (например, библиотеку стандартных функций) и другие системные программы.

9. Отладка и тестирование программы

Отладка – это устранение синтаксических и смысловых ошибок, которые могли быть допущены на всех этапах разработки программы. Синтаксические ошибки обнаруживают и исправляют, выполняя пробную трансляцию программы. Логические ошибки можно

выявить в процессе проверки программы на специально подобранных наборах входных данных. Поэтому тестирование определяют как выполнение программы в целях обнаружения скрытых ошибок. Ряд наборов исходных данных задаются такими, чтобы выполнялись все операторы (команды Ассемблера) и все ветви программы.

Тестированию уделяют в настоящее время большое внимание. «Работающая» программа проявит свои ошибки после длительного времени эксплуатации, если тестирование не было выполнено в полном объеме.

Завершая рассмотрение основных этапов решения задач на ЭВМ, заметим, что разделение этого процесса на 5 этапов носит условный характер. В случае простых программ некоторые этапы сливаются, например, проектирование с разработкой алгоритма или кодирование алгоритма с отладкой. В случае сложных программ могут добавиться новые этапы, например, проектирование базы данных или разработка интерфейса пользователя. При разработке сложной программы в процессе тестирования могут быть обнаружены ошибки, для исправления которых необходимо пересмотреть все этапы, возможно, и саму постановку задачи.

10. Выводы

Проанализировав самые востребованные учебники по программированию на С++, приходим к выводу, что в них отсутствуют, или освещены очень скупо темы, дающие ключ к пониманию самого процесса программирования. Рассмотрение основных конструкций языка, классов следует начинать лишь после того, как станут понятными этапы программирования. Последующими темами должны быть «Основы алгоритмизации» и «Типовые алгоритмические структуры».

Новизна и практическая значимость данной работы определяется следующим. Многолетний опыт преподавания дисциплины «Основы программирования» дает основания указать ту последовательность в изложении материала, при которой достигается наилучшее восприятие дисциплины, которая часто у обучающихся вызывает значительное затруднение в усвоении излагаемого материала.

В работе [5] показано, что алгоритм любой сложности может быть реализован лишь с использованием типовых алгоритмических структур, которые комбинируются различным образом в сложном алгоритме. Глубокое рассмотрение этих тем поможет обучающимся постичь самый сложный и ключевой этап в программировании – составление алгоритма. Эти темы планируется рассмотреть в сообщении 2 этой статьи.

Список литературы: 1. *Шилдт Г.* С++: базовый курс. 3-е издание. М.: Издательский дом «Вильямс», 2006. 624с. **2.** *Прата С.* Язык програмирования С++. Лекции и упражнения. 5-е издание. М.: ООО «И.Д. Вильямс», 2007. 1184 с. **3.***Лафоре Р.* Объектно-ориентированное программирование в С++. Классика Computer Science. 4-е издание. СПб.: Питер, 2005. 924с. **4.** *Дейтел Х.М., Дейтел П. Дж.* Как программировать на С++. 5-е малое издание. М.: ООО «Бином-Пресс», 2007. 800с. **5.** *Bohm C., Jacopini G.* Flow diadrams, Turing machines and languages with only twofomation rules. Communications of the ACM. 1966. Vol.9. P. 366-371. **6.** *Савитч У.* Язык С++. Курс объектно-ориентированного программирования. 3-е издание. М.: Издательский дом «Вильямс», 2001. 704 с.

Поступила в редколлегию 09.03.2010

Марченко Юрий Сергеевич, канд. техн. наук, доцент кафедры программного обеспечения ЭВМ ХНУРЭ. Научные интересы: алгоритмические языки и программирование. Адрес: Украина, 61166, Харьков, пр. Ленина, 14, тел. 8(057)7021446, дом. тел. 8(057)7026005.