М. И. Жадан, Е. М. Березовская
г. Гомель, ГГУ имени Ф. Скорины

СТИЛЬ ПРОГРАММИРОВАНИЯ И ВОЗМОЖНОСТИ ПРОГРАММИСТА

Какой должна быть программа? Чем отличаются программы, написанные студентами по выданному заданию? Какие программы хотел бы видеть преподаватель по выполняемому студентом заданию? Казалось бы, эти вопросы нелепы, так как студент, получая задание, уже имеет информацию о том, в каком виде она должна быть представлена преподавателю. Однако не все здесь так просто. Следует осознать, что программу можно обсуждать, критиковать, улучшать и сравнивать с другими программами, что она может быть «хорошей» или «плохой» и что важно сделать ее хорошей относительно некоторых критериев. Далее остановимся на некоторых из этих критериев.

Основным качеством программы, которое, в конечном счете, позволяет ее принять является «правильность»: всякая программа составляется для того, чтобы отвечать некоторому требованию и должна ему удовлетворять. Программа, которая «почти работает», как правило, вообще не работает у пользователя, который с ее помощью может получать ложные или случайные результаты. Проблема правильности программы трудно разрешима еще и потому, что нужно уметь определить требования, которым должна она удовлетворять. Это является не всегда простой задачей.

Достаточно часто на вопрос о проверке (тестировании) программы от студентов можно слышать, что при их данных программа работает успешно, а если нет, то что-то сделали не так. Тестирование – это простейшее для программиста действие по подтверждению правильности программы, но не всегда окончательное. По словам Дейкстры, «тесты могут служить для демонстрации наличия ошибок, но не их отсутствия». Конечно, никто бы не выпустил, например, программу управления движением поездов в метро, не протестировав ее с помощью моделирования. Это не означает, что испытания, выполненные для оценки правильности программы, не должны быть произвольными, если необходимо получить ценные сведения. Не следует ожидать многого от тестов, если они не были предусмотрены при написании программы. Если тесты присутствуют, то они являются заранее предусмотренным этапом разработки. Следует также отметить, что тесты могут служить важным средством оценки эффективности программы в тех случаях, когда математический анализ ее сложности является слишком громоздким [1].

Вопросы читаемости программы и ее результатов также являются важными в квалификационной характеристике программиста. Тот, кто пробовал погрузиться даже в свою программу, написанную несколько месяцев/лет назад, знает, что повторное ее чтение превращается в некоторую пытку, если проблеме чтения не было уделено должного внимания. Такую программу, возможно, легче написать с нуля, учитывая соответствующие изменения.

Легко читаемая программа позволяет читателю ее понять достаточно быстро. Следует отметить, что ключом к читаемости программы являются не нагромождения объяснений, а сама программа. Надо найти как можно проще и естественнее формы выражения, позволяющие установить читателю непосредственное соответствие между текстом программы и тем, что происходит при ее выполнении. Как правило, оценка стиля программы является субъективной, однако приведем некоторые принципы, применяемые в программировании:

– необходимо как можно реже использовать такие операторы, как Goto, Break, Continue, Exit;

– использование осмысленных имен. Использование таких имен, как «A», «AA», «AAA» является мазохистской практикой, поскольку возможна либо опечатка, либо ошибка;

– необходимо использовать смещения для выделения управляющих структур и их размещения;

– использовать «полезные» комментарии.

Общей чертой всех комментариев является то, что они должны быть составными частями программ, задуманными и построенными одновременно с этими программами.

Посмотрим на примере, как используются описанные выше методики составления программ в курсе «Информатика».

Пример. Определить, является ли введенный текст числом.

Решение 1:

```
program Chislo;
var
  s: string;
  i, flag: integer;
begin
  writeln('Введите строку: ');
  readln(s);
  flag := 1;  //Текст будет числом
  for i := 1 to Length(s) do
     if (pos(s[i], '1234567890') = 0) then
       begin flag := 0; break; end;
  if (flag = 1) then writeln('Данный текст является числом')
      else  writeln('Данный текст не является числом');
    end.
```

Протокол решения задачи:

```
Введите строку:                    Введите строку:
50246                              50к246ва
Данный текст является числом            Данный текст не является числом
```

Решение 2:

```
program Chislo;
var
  s: string;
  i, flag: integer;
begin
  writeln('Введите строку: ');
  readln(s);
  flag := 1;  //Текст будет числом
  for i := 1 to Length(s) do
     if (pos(s[i], '1234567890') = 0) then  flag := 0;
  if (flag = 1) then  writeln('Данный текст является числом')
      else   writeln('Данный текст не является числом');
end.
```

В этом случае протокол решения задачи будет такой же, как в решении 1.

Решение 3:

```
program Chislo;
var
  s: string;
  i, flag: integer;
begin
  writeln('Введите строку: ');
  readln(s);
  for i := 1 to Length(s) do
     if (pos(s[i], '1234567890') = 0) then  flag := 0
 else flag := 1;
  if (flag = 1) then  writeln('Данный текст является числом')
      else  writeln('Данный текст не является числом');
end.
```

Этот вариант решения содержит логическую ошибку, хотя первый тест из решения 1 был успешным. Ниже приведен тест, указывающий на эту ошибку.

```
Введите строку:
12d235
Данный текст является числом
```

В заключение отметим, что решение 1 содержит «нежелательный» оператор *Break*, с другой стороны, он позволил прервать процесс проверки в случае первого нахождения не числового элемента. Следует отметить, что такой же результат можно получить с использованием «нежелательного» оператора *Goto*. Решение 2 является логически структурированным, в то время как похожее решение 3 содержит логическую ошибку.

Литература

1. Мейер, Б. Методы программирования: в 2 т. / Б. Мейер, К. Бодуэн. – М.: Мир, 1988. – Т.2. – 368 с.