Е. М. Березовская, М. И. Жадан
Гомельский государственный университет имени Ф. Скорины

ОБ УПРАВЛЯЮЩИХ СТРУКТУРАХ И СТРУКТУРНОМ ПРОГРАММИРОВАНИИ

Как известно, решение задачи на ЭВМ требует ее постановки, анализа, моделирования построения алгоритма (например, в виде блок-схемы) и написания программы для получения численных расчетов. Качество программы зависит от квалификации программиста. Почти всегда программист свою программу считает «хорошей», хотя в ней могли быть использованы такие управляющие структуры как *GOTO* или *BREAK*.

Еще в 1966 году С. Боэм и Г. Якопини [1] доказали теорему о том, что для всякой программы, выраженной произвольной блок-схемой,

РЕПОЗИТОРИЙ ГГУ ИМЕНИ Ф. СКОРИНЫ

существует эквивалентная ей программа (т.е. выполняющая те же преобразования: данные – результат, с помощью тех же вычислений), так что:

1) операции над переменными те же, что и в исходной программе;

2) сохраняются все переменные исходной программы с возможным добавлением некоторого числа логических переменных;

3) единственными используемыми управляющими структурами являются: цепочка; цикл *пока* (While).

Эта теорема является теоретическим обоснованием для введения управляющих структур, но не причиной, по которой рекомендуется их исключительное использование. Такими причинами могут быть средства написания алгоритмов, которые представляют эти структуры, улучшают читаемость программ, облегчают общение между программистами и делают возможным доказательство некоторых свойств программ. Последнее отвечает целям, на которые ориентирован метод декомпозиции сложных задач, называемый структурным программированием [2].

Конечно, улучшение читаемости программы позволяет понимать их при чтении сверху вниз так, что текст программы моделирует, насколько возможно, их выполнение. Общение между программистами – это один из определяющих элементов программистского проекта. Более или менее строгое доказательство правильности программы является в итоге единственным средством убедиться в этом самому.

Таким образом, надо с самого начала пытаться выразить программируемую задачу с помощью примитивных структур типа «цепочка – пока», определяющих ясный и простой способ мышления и стараться сохранить его в ходе всех последовательных этапов разработки программы. Поэтому необходимо стараться исключить операторы перехода типа *GOTO*, которые мешают последовательному чтению программы.

Для пояснения сказанного приведем следующий пример.

Дан некоторый текст.

Необходимо выдать сообщение, если текст состоит только из малых английских букв, в противном случае – выдать первый не английский символ.

Решение 1: Ниже приведено решение, содержащее оператор *GOTO*.

```
program Malangl;
  const angl=['a'..'z'];
var
  s: string;
```

РЕПОЗИТОРИЙ ГГУ ИМЕНИ Ф.СКОРИНЫ

```
  i,n, flag: integer;
  label M;
begin
  writeln('Введите строку: ');
  readln(s);
  flag := 1; //текст будет содержать только англ малые буквы
  n:=Length(s);  //длина текста
  for i := 1 to n do
   if NOT(s[i] in angl) then
     begin
       flag := 0;
       goto M;
     end;
  M:if (flag = 1) then
     writeln('текст содержит только англ малые буквы')
    else begin
       writeln('текст содержит не только англ малые буквы');
       writeln('первый не англ символ=',s[i]);
        end;
end.
```

Протокол выполнения программы имеет вид:

1) Введите строку:

dfdg5

текст содержит не только англ малые буквы

первый не англ символ=5

2) Введите строку:

dgfhjb

текст содержит только англ малые буквы

Решение 2: Это решение не содержит оператора *GOTO* и является более лаконичным. Протокол выполнения программы такой же, как в решении 1.

```
program Malangl;
  const angl=['a'..'z'];
var
  s: string;
  i,n,k,flag: integer;
begin
  writeln('Введите строку: ');
  readln(s);
  flag := 1; //текст будет содержать только англ малые буквы
```

РЕПОЗИТОРИЙ ГГУ ИМЕНИ Ф. СКОРИНЫ

```
n:=Length(s);  //длина текста
i := 1;
while (i<= n) do
  if not(s[i] in angl) then
    begin
      flag := 0; k:=i; i:=n+1;
    end else i:=i+1;
  if (flag = 1) then
   writeln('текст содержит только англ малые буквы')
  else begin
    writeln('текст содержит не только англ малые буквы');
    writeln('первый не англ символ=',s[k]);
     end;
end.
```

Следующий фрагмент программы слева, выводящий число 543210, показывает, как не надо писать программы. Справа реализован его эквивалент.

```
    label 1,2;
begin                                   for i:= 5 downto 0 do
    var i := 5;                             write(i);
2: if i<0 then goto 1;
  write(i);
  Dec(i);
  goto 2;
1:
end.
```

Использование оператора безусловного перехода в программе считается признаком плохого стиля программирования. Один из немногих примеров уместного использования оператора *GOTO* в программе – это выход из нескольких вложенных циклов одновременно.

РЕПОЗИТОРИЙ ГГУ ИМЕНИ Ф.СКОРИНЫ

Список использованной литературы

1. Boehm, C. Flow Diagrams, Turing Machines, and Languages With Only Two Formation Rules / C. Boehm. – CommACM, 1966. –366-371 p.

2. Дал, У. Структурное программирование / У. Дал. – 235 с.– М.: Мир, 1975.