электронный научно-технический журнал
ИНЖЕНЕРНЫЙ ВЕСТНИК
Издатель ФГБОУ ВПО "МГТУ им. Н.Э. Баумана". Эл No. ФС77-51036. ISSN 2307-0595

Методические приемы разработки программ с использованием фрагментов типовых алгоритмов

 # 05, май 2014

доцент, к.т.н. Ваулин А. С.

УДК: 004.043.085

Россия, МГТУ им. Баумана
vaulin.1935@mail.ru

Введение

Использование языка Паскаль для обучения программированию связано с тем, что он позволяет на начальной стадии освоить работу в консольном режиме (принципы структурного программирования) [1], а затем перейти к графическому режиму (объектно-ориентированное программирование) [2- 6].

В консольном режиме не требуются дополнительные знания кроме языка программирования и можно сосредоточить внимание на разработке программ с простым и наглядным интерфейсом.

На подготовку таких программ в целом требуется меньше времени, поэтому консольный режим удобно использовать для быстрой проверки и отладки отдельных задач. Переход от редактирования программы к исполнению осуществляется нажатием быстрой кнопки панели инструментов или клавиши F9.

Изучение дисциплины «Информатика» расширяет алгоритмическое мышление и способствует дальнейшему использованию вычислительной техники в общеинженерных и специальных дисциплинах.

Многие студенты в начальной стадии обучения испытывают психологический барьер при разработке программ. Это связано с необходимостью восприятия большого объема формализованных понятий, типов данных, конструкций языка, структур данных и др. Поэтому, пока твердо не усвоены основные понятия языка, предлагается использовать заранее отлаженные фрагменты типовых алгоритмов. В этом случае студент сосредотачивает внимание на формировании программы, выбирая необходимые фрагменты программ, и затем, включая эти фрагменты программ в нужной последовательности.

Приступая к программированию с использованием типовых фрагментов программ, студент должен иметь знания о структуре программы, описаниях данных, операторах ввода-вывода данных, основных операторах языка. Эти начальные знания должны дать воз-

можность выбора необходимых фрагментов из возможного набора типовых фрагментов программ.

Решение задач требует наличия достаточных знаний, чтобы понять существо задания и выбрать под данное задание необходимые фрагменты программ. Студенту должны быть доступны фрагменты программ из разделов описаний, фрагменты ввода исходных значений, фрагменты обработки данных и фрагменты вывода результатов обработки данных. Следовательно, используя фрагменты программ, студент компонует программу по алгоритму линейной структуры: описания исходных данных – ввод данных – обработка данных – вывод результатов обработки.

Структура программы и описания данных

При входе в консольный режим появляется окно приложения с шаблоном в центре окна в виде:

program Project # N; // заголовок программы, # N- номер проекта (приложения)
{$APPTYPE CONSOLE} // директива консольного режима работы
 (uses SysUtils;) // подключение модулей, системой и пользователем
begin // раздел операторов, заключенный в операторные скобки
{ TODO -oUser -cConsole Main : Insert code here }
end. // **.** (точка) указывает компилятору о конце программы

Внутри раздела операторов находится комментарий (поясняющий текст), предлагающий разместить в этом месте операторы, реализующие алгоритм программы. Комментарии, в отличие от операторов, не порождают команд процессора при обработке компилятором и он их просто пропускает. Назначение комментариев состоит в том, чтобы облегчить понимание отдельных функциональных частей программы и алгоритма программы в целом.

Используются одно строчные комментарии, начинающиеся с символа // и много строчные, текст которых заключается в фигурные скобки { и } или в скобки вида (* и *).

Комментарии в начальной стадии программирования нужно использовать не только для отдельных функциональных частей программы, но и для описаний обрабатываемых данных и отдельных операторов.

Перед разделом операторов размещаются разделы описаний, которые не показаны в шаблоне программы. Язык программирования является языком с обязательным описанием всех обрабатываемых данных и включает следующее разделы описаний [7-17]:

- Label (раздел описания меток),
- Const (раздел описания констант),
- Type (раздел описания типов),
-Var (раздел описания переменных и массивов),
- Procedure (раздел описания процедур),
- Function (раздел описания функций).

Разделы могут использоваться в любом порядке и при необходимости могут повторяться в описании. По правилам языка используемому объекту должно предшествовать его описание. При отсутствии, какого – либо раздела в описании он опускается.

Программа решения задачи представляет собой логически связанные части, каждая из которых выполняет отдельную функциональную часть алгоритма задачи.

Совокупность обрабатываемых данных в программе состоит из подмножеств данных общих для нескольких частей программы и подмножеств, использующихся только в пределах отдельной части программы.

При разработке программы перед студентом возникают задачи, как выделить функциональные части и описать их средствами языка, как выразить средствами языка последовательности выполняемых действий.

Имеются различные средства установления отношений между частями разрабатываемой программы (блочный, процедурный, модульный). При разработке простых программ достаточно использовать блочный принцип. Используя блочный принцип можно подготовить типовые фрагменты:

- описаний данных,
- ввода данных,
- обработки данных,
- вывода результатов.

Предлагаемая методика будет демонстрироваться на ограниченных примерах обработки одномерных и двумерных массивов данных, используя типовые фрагменты алгоритмов.

В соответствии с выделенными функциональными частями программ необходимо подготовить типовые описания констант, переменных и массивов. Разрабатываемые фрагменты должны учитывать основные свойства алгоритмов и программ - массовость и результативность.

Эти принципы должны закладываться, начиная с описания данных. При выборе имен переменных и массивов нужно выбирать имена, отражающие физический смысл, функциональную направленность и логическую связь между собой.

В разделе описаний константой задается максимально возможное значение числа элементов. Например, при описании одномерного массива A его максимальный размер может быть задан в разделе констант в виде nmaxA=50;. Эта константа будет использована при описании массивов. Учитывая свойство массовости при обработке массива, необходимо задать реально обрабатываемое число элементов. Например, для массива A число элементов nA.

После выбора фрагмента и переноса его в программу нужно вставить подробные комментарии к функциональным частям и отдельным операторам

Типовые фрагменты ввода должны содержать простейший интерфейс, помогающий быстрее понять, что необходимо вводить и в какой последовательности.

Фрагменты типовых операций ввода данных могут быть реализованы в следующем виде. Для одномерных массивов необходимо задать реально обрабатываемое число элементов, затем организовать ввод элементов массива. Для наглядности ввод отдельных данных нужно отделять пустой строкой. Простейший интерфейс помогает понять, какие необходимые действия нужно выполнить.

Фрагмент ввода данных для одномерного массива примет вид

```
Writeln ('Введите число элементов массива A');
Readln (nA);
Writeln; // пустая строка разделяет данные ввода
Writeln ('Введите значения элементов массива A');
For i: =1 to nA do
   Read (A[i]);
Writeln; // пустая строка разделяет данные ввода от результатов обработки
```

Аналогично выполняется ввод двумерного массива (матрицы), вводятся реально обрабатываемое число строк и число столбцов, а затем во вложенном цикле осуществляется ввод элементов матрицы. Нагляднее ввод матрицы выполнить в естественном виде, вводя элементы матрицы построчно.

Фрагменты типовых алгоритмов обработки включают в себя наиболее часто встречающиеся вычисления различных сумм и произведений, нахождение наибольших (наименьших) значений, различного вида сортировки и другие [7-11].

Фрагменты операций вывода результатов обработки должны выводить результаты простых переменных со своими именами в виде <имя переменной>=<значение>. Значения действительного типа должны выводиться в формате в виде целой и дробной части, а для элементов массива предварительно выводится заголовок с именем массива.

Следовательно, имея определенный класс решаемых задач можно подготовить различные фрагменты программ.

1. Типовые фрагменты описаний переменных и массивов

1.1 Раздел описания констант (const)

1.1.1 Максимально возможное число элементов массива

```
mmaxA=50;
```

1.1.2. Максимально возможное число строк и столбцов матрицы

```
mmaxB=12; nmaxB=14;
```

1.2. Раздел описания переменных и массивов (var)

1.2.1. Реально обрабатываемое число элементов массива A

```
nA: integer;
```

1.2.2. Реально обрабатываемое число строк и столбцов матрицы B

```
mB, nB: integer;
```

1.2.3. Наибольший (наименьший) элемент массива A

```
Amax: real; (Amin: real;)
```

1.2.4. Сумма положительных элементов массива A

sump: real;

1.2.5. Количество положительных элементов массива A

kp: integer;

1.2.6. Параметры циклов

i: integer; //для простого цикла или для внешнего вложенного цикла

j: integer; // для внутреннего вложенного цикла

1.2.7. Среднеарифметическое положительных элементов массива A

srA: real;

1.2.8. Индекс наибольшего (наименьшего) элемента массива A

imaxA:integer; (iminA: integer;)

1.2.9. Массив элементов A действительного типа

A: array [1..nmax] of real;

1.2.10. Матрица B действительного типа

B: array [1..mmax,1..nmax] of real;

1.2.11. Массив Bmax наибольших элементов строк матрицы B

Bmax: array [1..mmax] of real;

2. Типовые фрагменты операций ввода

2.1. Ввод простых переменных

2.1.1. Ввод числа элементов обрабатываемого массива A.

```
Writeln ('Введите число элементов массива A');
Readln (nA);
```

2.1.2. Ввод числа строк и столбцов матрицы B.

```
Writeln ('Введите число строк и столбцов матрицы B');
Readln (mB, nB);
```

2.2 Ввод значений элементов массива

2.2.1. Ввод значений элементов массива A.

```
Writeln ('Введите значения элементов массива A');
For i: =1 to nA do
    Read (A[i]);
```

2.2.2. Ввод значений элементов значений элементов матрицы B в виде матрицы.

```
Writeln ('Введите значения элементов матрицы B ');
For i: =1 to mB do
begin
    For j: =1 to nB do
        Read (B[i,j]);
        Readln
end;
```

3. Типовые фрагменты алгоритмов обработки

3.1. Вычисление суммы и числа положительных элементов массива A

```
sump:=0;
kp:=0;
For i:=1 to n do
   If A[i] >0 then begin sump:= sump + A[i]; kp:= kp +1 end;
```

3.2. Нахождение наибольшего (наименьшего) значения и его порядкового номера среди элементов массива A

```
Amax:= A[1]; //для сравнения берется первый элемент массива
imax:=1;
For i:=2 to nA do
   If A[i]> Amax then begin Amax :=  A[i]; imax:= i end;
```

3.4. Нахождение наибольшего значения среди элементов каждой строки матрицы B и сохранение их в массиве Bmin.

```
For i: =1 to mB do
   begin
      Bmax[i]:= B[i,1]; // для сравнения берется первый элемент строки массива
      For j: =2 to nB do
         If B[i,j]]> Bmax[i] then begin Bmax[i]  :=  B[i,j] end;
   end;
```

4. Типовые фрагменты операций вывода результатов обработки

4.1. Вывод простых переменных

4.1.1. Вывод суммы и количества положительных элементов массива A

```
Writeln ('Сумма положительных элементов sump=', s:6:2);
Writeln ('Число положительных элементов kp=', k:2);
```

4.1.2. Вывод наибольшего значения и его индекса

```
Writeln (Наибольшее значение Amax =', Amax:6:2);
Writeln ('Номер наибольшего значения imax =', imax:2);
```

4.2. Вывод массивов

4.2.1. Вывод значений элементов массива A

Выводится заголовок к значениям массива A, значения элементов массива выводятся в формате действительных чисел (с двумя цифрами в дробной части и тремя пробелами между значениями массива)

```
Writeln ('Элементы массива A');       // заголовок к элементам массива
For i: =1 to nA do
   Write (A[i]:6:2, ' ':3);   // значения выводятся в одной строке
```

4.2.2. Вывод значений элементов матрицы B в виде матрицы

```
Writeln ('Элементы матрицы B ');
For i: =1 to mB do //внешний цикл для выбора индекса строки
```

```
begin
    For j: =1 to nB do //внутренний цикл  для выбора индекса столбца
        Write (B[i,j]:6:2, ' ':3);  //вывод элементов строки матрицы
        Writeln;  //переход на следующую строку матрицы
end;
```

4.2.3. Вывод наибольших значений каждой строки матрицы B

```
Writeln ('Наибольшие значения каждой строки матрицы B');
For i:=1 to mB do
    Read (Bmin[i]:6:2, ' ':3);
Writeln;
```

Примеры использования типовых фрагментов программ

Задание 1.

Вычислить сумму sump и число положительных элементов kp массива A.

Вывести значения элементов массива A и полученные результаты с поясняющим текстом.

Шаблон программы заполняется в соответствии с последовательностью действий: описание переменных и массивов – ввод входных данных – обработка данных – вывод результатов обработки. После ввода элементов массива A их необходимо вывести для наглядности обрабатываемых данных и их контроля. Окончательная последовательность действий примет вид: ввод входных данных – вывод входных данных – обработка данных – вывод результатов обработки.

Из предложенных описаний выбираются описание константы mmaxA (1.1.1), описания переменных nA (1.2.1), sump (1.2.4), kp (1.2.5), i (1.2.6), массива A (1.2.9).

Для ввода данных выбираются фрагменты 2.1.1 и 2.2.1 , для вывода элементов массива A фрагмент (4.2.1), для вычисления суммы и числа положительных элементов фрагмент 3.1 и функций вывода фрагмент 4.1. 1.

Для наглядности входные данные и результаты обработки нужно отделять друг от друга пустой строкой.

На начальном этапе программирования при заполнении шаблона фрагментами программ необходимо записывать комментарии, выделяя функциональные части программы и поясняя описания переменных и назначения отдельных операторов.

В результате выполненных действий программа принимает следующий вид.

```
program Project 1;
{$APPTYPE CONSOLE}
 (uses SysUtils;)
Const
    mmaxA=50; // максимально возможное число элементов массива
var
```

```
    nA: integer; // реально обрабатываемое число элементов массива
    sump: real; // переменная для накапливания суммы положительных элементов масси-
ва
    kp: integer; // переменная для накапливания числа положительных элементов массива
    i: integer;  // параметр цикла
    A: array [1..nmax] of real; //массив действительных значений
begin // раздел операторов, заключенный в операторные скобки
{ TODO -oUser -cConsole Main : Insert code here } //
//Ввод данных
    Writeln ('Введите число элементов массива A');
    Readln (nA);
    Writeln; //пустая строка разделяет данные
    Writeln ('Введите значения элементов массива A');
    For i: =1 to nA do
      Read (A[i]);
    Writeln; //пустая строка разделяет данные ввода от результатов обработки
    // Вывод элементов массива A
    Writeln ('Элементы массива A');       // заголовок к элементам массива
    For i: =1 to nA do
      Write (A[i]:6:2, ' ':3);   // значения размещаются в одной строке
    Writeln; //пустая строка разделяет данные ввода от результатов обработки
    // Вычисление суммы и количества положительных элементов
    sump:=0; //начальное значение счетчика суммы положительных элементов
    kp:=0; //начальное значение счетчика числа положительных элементов
    For i:=1 to n do
      If A[i] >0 then begin sump:= sump+ A[i]; kp:=kp+1 end;
    //вывод
    Writeln ('Сумма положительных элементов sump=', s:6:2);
    Writeln ('Число положительных элементов kp=', k:2);
    Readln; //задержка закрытия окна программы до нажатия клавиши Enter
end.
```

Задание 2.

Найти наибольшее значение и его порядковый номер среди элементов массива A. Полученные результаты вывести.

Шаблон программы заполняется аналогично заданию 1 в соответствии с постановкой задачи.

Из предложенных описаний выбираются описание константы mmaxA (1.1.1), описания переменных nA (1.2.1), Amax (1.2.3), i (1.2.6), imax (1.2.8), массива A (1.2.9).

Для функций ввода выбираются фрагменты 2.1.1 и 2.2.1 , для вывода элементов массива А фрагмент (4.2.1), для нахождения наибольшего значения и его порядкового номера фрагмент 3.2 и функций вывода фрагмент 4.1.2.

```
program Project 2;
{$APPTYPE CONSOLE}
 (uses SysUtils;)
Const
      mmaxA=50;
var
      nA: integer;
      sump: real;
      kp: integer;
      i: integer;
      A: array [1..nmax] of real;
begin // раздел операторов, заключенный в операторные скобки
{ TODO -oUser -cConsole Main : Insert code here } //
      //Ввод данных
      Writeln ('Введите число элементов массива А');
      Readln (nA);
      Writeln; //пустая строка разделяет данные
      Writeln ('Введите значения элементов массива А');
      For i: =1 to nA do
         Read (A[i]);
      Writeln; //пустая строка разделяет данные ввода от результатов обработки
      // Вывод элементов массива А
      Writeln ('Элементы массива А');       // заголовок к элементам массива
      For i: =1 to nA do
         Write (A[i]:6:2, ' ':3);   // значения размещаются в одной строке
      Writeln; //пустая строка разделяет данные ввода от результатов обработки
      Amax:= A[1]; //для сравнения берется первый элемент массива
      imax:=1;
      For i:=2 to nA do
         If A[i]> Amax then begin Amax :=  A[i]; imax:= i end;
      Writeln (Наибольшее значение Amax =', Amax:6:2);
      Writeln ('Номер наибольшего значения imax =', imax:2);
      Readln; //задержка закрытия окна программы до нажатия клавиши Enter
end.
```

Задание 3.

Найти наибольшие значения элементов каждой строки матрицы B и сохранить их в массиве Bmax. Вывести элементы матрицы B в виде матрицы и элементы полученного массива Bmax.

Из предложенных описаний выбираются описания констант mmaxB и nmaxB (1.1.2), описания переменных mB, nB (1.2.2), i, j (1.2.6), массивы B (1.2.10) и Bmax (1.2.11).

Для ввода числа строк и столбцов выбирается фрагмент (2.1.2), а для ввода элементов матрицы – фрагмент (2.2.2) , для вывода элементов матрицы B фрагмент (4.2.2), для нахождения наибольших значений элементов строк матрицы B фрагмент (3.4) и для вывода элементов массива Bmax фрагмент (4.2.3).

```
program Project 3;
{$APPTYPE CONSOLE}
 (uses SysUtils;)
Const
    mmaxB=12; nmaxB=14;
var
    mB, nB: integer;
    i, j: integer;
    B: array [1..mmax,1..nmax] of real;
    Bmax: array [1..mmax] of real;
begin // раздел операторов, заключенный в операторные скобки
{ TODO -oUser -cConsole Main : Insert code here } //
//Ввод данных
    Writeln ('Введите число строк и столбцов матрицы B');
    Readln (mB, nB);
    Writeln;
    Writeln ('Введите значения элементов матрицы B ');
    For i: =1 to mB do
    begin
       For j: =1 to nB do
          Read (B[i,j]);
       Readln
    end;
    Writeln ('Элементы матрицы B ');
    For i: =1 to mB do //внешний цикл для выбора индекса строки
    begin
       For j: =1 to nB do //внутренний цикл  для выбора индекса столбца
          Write (B[i,j]:6:2, ' ':3);  //вывод элементов строки матрицы
       Writeln;  //переход на следующую строку матрицы
    end;
    Writeln;
```

```
// Вывод элементов матрицы B
Writeln ('Элементы матрицы B ');
For i: =1 to mB do //внешний цикл для выбора индекса строки
begin
    For j: =1 to nB do //внутренний цикл  для выбора индекса столбца
        Write (B[i,j]:6:2, ' ':3);  //вывод элементов строки матрицы
    Writeln;  //переход на следующую строку матрицы
end;
Writeln;
// нахождение  наибольших значений элементов строк
For i: =1 to mB do
begin
    Bmax[i]:= B[i,1]; // для сравнения берется первый элемент строки массива
    For j: =2 to nB do
        If B[i,j]]> Bmax[i] then begin Bmax[i]  :=  B[i,j] end;
end;
// Вывод  наибольших значений элементов строк
Writeln ('Наибольшие значения каждой строки матрицы B');
For i:=1 to mB do
    Read (Bmax[i]:6:2, ' ':3);
Readln; //задержка закрытия окна программы до нажатия клавиши Enter
end.
```

Заключение

Использование типовых фрагментов программ в начальной стадии обучения позволяет исключить синтаксические ошибки в записи конструкций языка, сократить время отладки программ и получения результатов обработки, в значительной степени снять психологический барьер и повысить интерес к обучению..

Список литературы

1. Алексеев В.Е., Ваулин А.С., Куров А.В. Практикум по программированию. Обработка числовых данных. М.: Изд-во МГТУ им. Баумана, 2008, 285
2. Морун А.Н. Программирование на языке Паскаль (Pascal). Основы обработки структур данных. М.: Диалектика, 2005, 576
3. Шелест В.П. Программирование. СПб.: БХВ - Петербург, 2002, 243
4. Рапаков Г.Г., Ржеуцкая С.Ю. Программирование на языке Pascal. СПб.: БХВ – Петербург, 2004, 480
5. Культин Н.Б. Основы программирования в Turbo Delphi. СПб.: БХВ - Петербург, 2007, 384

6. Культин Н.Б. Программирование в Turbo Pascal 7.0 и Delphi. СПб.: БХВ - Петербург, 2012, 390
7. Алексеев В.Е., Ваулин А.С., Петрова Г.Б. Вычислительная техника и программирование. Практикум по программированию. Под ред. А.В. Петрова. М.: Высшая школа, 1991, 400
8. Вычислительная техника и программирование /А.В. Петров, В.Е. Алексеев, А.С. Ваулин, Г.Б. Петрова, М.А. Титов, П.Н. Шкатов; учебник под ред. А.В. Петрова. М.: Высшая школа, 1990, 479
9. Инструментальные средства ПЭВМ. В 10 кн. Кн. 4. Программирование в среде Турбо ПАСКАЛЬ /Л.Е. Агабеков, С.В. Борисов, А.С. Ваулин, А.Д.Козлов, А.В. Куров, И.Л. Серебрякова; под ред. Б.Г. Трусова. М.: Высшая школа, 1993, 142
10. ЭЛЕКТРОННЫЕ ВЫЧИСЛИТЕЛЬНЫЕ МАШИНЫ. В 8 кн. Кн. 7. Практикум по программированию /В.Е. Алексеев, А.С. Ваулин; под ред. А.Я. Савельева. М.: Высшая школа, 1993, 207
11. ЭЛЕКТРОННЫЕ ВЫЧИСЛИТЕЛЬНЫЕ МАШИНЫ. В 8 кн. Кн. 5. Языки программирования (ПАСКАЛЬ, ПЛ/М) /А.С. Ваулин; под ред. А.Я. Савельева. М.: Высшая школа, 1993, 157
12. Фаронов В.В. Программирование на персональных ЭВМ в среде ТУРБО-ПАСКАЛЬ. М.: Изд-во МГТУ им. Баумана, 1990, 580.
13. Васильев П.П. Турбо Паскаль в примерах и задачах. М.: Финансы и статистика, 2003, 496
14. Культин Н.Б. Turbo Pascal в задачах и примерах. СПб.: БХВ- Петербург, 2006, 256
15. Мельников С.П. Delphi и Turbo Pascal на занимательных примерах. СПб.: БХВ - Петербург, 2012, 448
16. Потомахин В.К. Turbo Pascal. Освой на примерах. СПб.: БХВ - Петербург, 2005, 240
17. Зеленяк О.П. Современный задачник по ТУРБО ПАСКАЛЮ. М.: Изд. Дом ДМК Пресс, 2014, 312