Информатика 9 А класс

https://digital.prosv.ru/ , https://media.prosv.ru/content/, https://media.prosv.ru/

Группа компаний «Просвещение», открывает свободный доступ к электронным формам учебников

Дистанционные уроки на неделю с 13 по 17 апреля 2020, 1 час в неделю

Учитель физики информатики Гаджиагаев Тагир Гаджиагаевич

Учебник Семакин Залогова Информатика 9 класс

Внимание! Ответы на вопросы и задания оформлять письменно в рабочих тетрадях. Работы будут проверены

1 занятие

§ 13 Программы ветвлений на Паскале , стр 86-90

Задание : Прочитать параграф, ответить на вопросы в конце параграфа

7. Составьте алгоритм, по которому на компьютере будет происходить следующее: в переменную S вводится возраст Саши, в переменную M вводится возраст Маши. В качестве результата на экран выводится фраза «Саша старше Маши» или «Маша старше Саши» (предполагаем, что кто-нибудь из них обязательно старше).
8. Решите предыдущую задачу, учитывая возможность одинакового возраста Саши и Маши. В таком случае может быть получен ответ: «Саша и Маша — ровесники».
9. Составьте алгоритм упорядочения значений трех переменных по возрастанию, т. е. при любых исходных значениях A, B, C отсортируйте их так, чтобы стало: $A \le B \le C$. Проверьте алгоритм трассировкой при разных вариантах значений исходных данных.

ЕК ЦОР: часть 2, глава 6, § 36. ЦОР № 6, 12–14.

§ 13

Программирование ветвлений на Паскале

Основные темы параграфа:

- *оператор ветвления на Паскале;*
- *программирование полного и неполного ветвления;*
- *программирование вложенных ветвлений;*
- *логические операции;*
- *сложные логические выражения.*

Оператор ветвления на Паскале

В языке Паскаль имеется **оператор ветвления**. Другое его название — условный оператор.

Формат полного оператора ветвления следующий:

```
if <логическое выражение> then <оператор1>
                          else <оператор2>
```

Здесь `if` — «если», `then` — «то», `else` — «иначе».

Программирование полного и неполного ветвления

Сравните запись алгоритма БИД1 из предыдущего параграфа с соответствующей программой.


```
алг БИД1
вещ A, B, C
нач
   ввод A, B
   если A>B
        то C:=A
        иначе C:=B
   кв
   вывод C
кон
```

```
Program BID1;
var A, B, C: real;
begin
  readln(A, B);
  if A>B
  then C:=A
  else C:=B;

  writeln(C)
end.
```

Очень похоже на перевод с русского языка на английский. Обратите внимание на следующее отличие: в программе нет специального знака конца оператора ветвления. Здесь принято, оставлять в программе пустую строку совсем не обязательно. Здесь это сделано только ради наглядности.)

Простой формой логического выражения является **операция отношения**. Как и в АЯ, в Паскале допускаются все виды отношений (ниже указаны их знаки):

< (меньше);	<= (больше или равно);
> (больше);	= (равно);
<= (меньше или равно);	<> (не равно).

А теперь запрограммируем на Паскале алгоритм БИД2, в котором использовано неполное ветвление.

```
алг БИД2
вещ A, B, C
нач
   ввод A, B
   C:=A
   если B>A
        то C:=B
   кв
   вывод C
кон
```

```
Program BID2;
var  A, B, C : real;
begin
  readln(A, B);
  C:=A;
  if  B>A
  then C:=B;

  write(C)
end.
```

Опять всё очень похоже. Ветвь `else` в операторе ветвления может отсутствовать.

Программирование вложенных ветвлений

Запишем на Паскале программу определения большего из трех чисел, блок-схема которой показана на рис. 2.6. Структура этого алгоритма — вложенные ветвления. Алгоритм на АЯ (БИТ2) приведен в предыдущем параграфе.

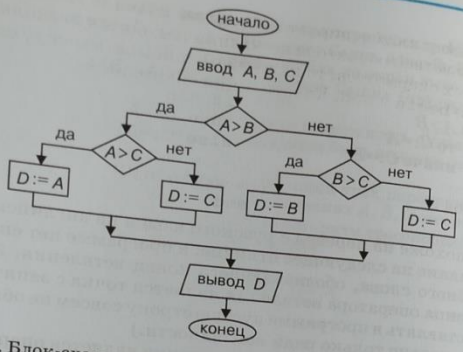

Рис. 2.6. Блок-схема алгоритма «БИТ» с вложенными ветвлениями

```
Program BIT2;
var A, B, C, D: real;
begin
   readln(A, B, C);
   if A>B
      then if A>C then D:=A else D:=B
      else if B>C then D:=B else D:=C;
   writeln(D)
end.
```

Обратите внимание на то, что перед **else** символ «;» не ставится, так как этот символ является разделителем операторов. Вся ветвящаяся часть структуры алгоритма заканчивается на точке с запятой после оператора `D:=C`.

Составим программу упорядочения значений двух переменных.

```
алг СОРТИРОВКА
вещ X, Y, C
нач
   ввод X, Y
   если X>Y
      то C:=X
         X:=Y
         Y:=C
   кв
   вывод X, Y
кон
```

```
Program SORTING;
var X, Y, C: real;
begin
  readln(X, Y);
  if  X>Y
  then begin C:=X;
             X:=Y;
             Y:=C
       end;
write(X,Y)
end.
```


Этот пример иллюстрирует следующее правило Паскаля: если на какой-то из ветвей оператора ветвления находится несколько последовательных операторов, то их нужно записывать между служебными словами **begin** и **end**. Конструкция такого вида:

begin <последовательность операторов> **end**

называется **составным оператором**. Следовательно, в описанной выше общей форме ветвления <оператор1> и <оператор2> могут быть простыми и составными операторами.

Логические операции

Наконец, составим еще один, третий вариант программы определения наибольшего числа из трех.

```
Program BIT3;
var A, B, C, D: real;
begin
      readln(A,B,C);
      if (A>=B) and (A>=C) then D:=A;
      if (B>=A) and (B>=C) then D:=B;
      if (C>=A) and (C>=B) then D:=C;
      writeln(D)
end.
```

Нетрудно понять смысл этой программы. Здесь использованы три последовательных неполных ветвления. А условия ветвлений представляют собой **сложные логические выражения**, включающие **логическую операцию and** (И). С логическими операциями вы встречались, работая с базами данных и с электронными таблицами.

Напомним, что операция **and** называется логическим умножением или конъюнкцией. Ее результат — «истина», если значения обоих операндов — «истина». Очевидно, что если $A \geq B$ и $A \geq C$, то A имеет наибольшее значение и т. д. В Паскале присутствуют все три основные логические операции:

and — И (конъюнкция),
or — ИЛИ (дизъюнкция),
not — НЕ (отрицание).

Замечание. Мы рассмотрели три варианта решения задачи о поиске максимального числа из трех заданных (БИТ1, БИТ2, БИТ3). Предложенные варианты решения, помимо всего прочего, отличаются еще и эффективностью: эффективность тем выше, чем меньше исполняется операций при выполнении программы. С этой точки зрения, самым неэффективным является алгоритм БИТ3. В любом случае в нем будет выполняться 6 операций отношения. При исполнении алгоритмов БИТ1 и БИТ2 выполняются 2 операции отношения. Понятно, что на практике необходимо использовать наиболее эффективные алгоритмы.

Сложные логические выражения

Обратите внимание на то, что отношения, связываемые логическими операциями, заключаются в скобки. Так надо делать всегда! Например, требуется определить, есть ли среди чисел A, B, C хотя бы одно отрицательное. Эту задачу решает следующий оператор ветвления:

```
if (A<0) or (B<0) or (C<0)
  then write('YES') else write('NO')
```

Выражение, истинное для отрицательного числа, может быть записано еще и так:

```
not (A>=0)
```

Коротко о главном

Формат оператора ветвления (условного оператора) Паскаля:

```
if <логическое выражение>
  then <оператор1> else <оператор2>
```

На ветвях условного оператора могут находиться простые или составные операторы. Составной оператор — это последовательность операторов, заключенная между служебными словами **begin** и **end**.

В сложных логических выражениях используются логические операции: **and**, **or**, **not**.

Вопросы и задания

1. Как программируется на Паскале полное и неполное ветвление?
2. Что такое составной оператор? В каких случаях составной оператор используется в операторе ветвления?
3. Выполните на компьютере все программы, приведенные в данном параграфе.
4. Составьте не менее трех вариантов программы определения наименьшего из трех данных чисел.
5. Составьте программу сортировки по возрастанию значений трех переменных: A, B, C.
6. Составьте программу вычисления корней квадратного уравнения по данным значениям его коэффициентов.

ЕК ЦОР: часть 2, глава 6, § 37. ЦОР № 6.

§ 14
Программирование диалога с компьютером

Основные темы параграфа:

- *что такое диалог с компьютером;*
- *пример программирования диалога.*

Что такое диалог с компьютером

Если вы исполняли рассмотренные выше программы на компьютере, то почувствовали определенное неудобство при работе с машиной. Во-первых, непонятно, когда машина начинает ожидать ввода данных, какие данные и в каком порядке нужно вводить (это ведь можно и забыть). Во-вторых, результаты получаются в виде чисел на экране, без всяких пояснений их смысла. Ясно, что люди между собой так не общаются.

Любую программу составлять нужно так, чтобы ее исполнение реализовывало диалог между компьютером и пользователем в понятной для человека форме.

Прежде чем начать составление программы, нужно продумать **сценарий** такого диалога.

Например, составим сценарий работы программы, вычисляющей сумму двух целых чисел. На экране компьютера последовательно должны появляться следующие строки (для примера предположим, что будем вводить числа 237 и 658):

Введите первое слагаемое: A = **237**
Введите второе слагаемое: B = **658**

A + B = 895
Пока!

Здесь курсивом записаны символы, которые выводит компьютер по программе, а прямым жирным шрифтом — символы, вводимые пользователем.

Любой вывод на экран происходит по оператору вывода, записанному в программе.