**ТЕМА: Программирование разветвляющихся алгоритмов**

**на языке Pascal.**

*Примечание*. Результат выполнения каждого упражнения для последующей его передачи по электронной почте преподавателю для проверки должен быть оформлен следующим образом:

1. Оформление выполнения задания осуществляется на компьютере с использованием текстового редактора.
2. Отправитель документа должен быть идентифицирован. Для этого в заголовке документа **должны быть указаны:** Индекс своей группы, Фамилия/Имя отправителя, Дата выполнения (отправления), Заголовок темы задания (см. Приложение 1)\*
3. Индекс и текст задания должны быть перепечатаны
4. Для оформления блок-схемы решения используется набор инструментов для рисования в тестовом редакторе
5. Текст программы на языке Pascal и результат выполнения программы для конкретно выбранных исходных данных отражается в ответе выполнением "скриншота" (использование кнопка клавиатуры $\frac{Prt}{SysRq}$ )

**\*Адрес получателя выполненных заданий: Логин komakhinav**

**Основные вопросы, рассматриваемые в рамках данной темы:**

1. Назначение и описание условного оператора.
2. Особенности структурного построения программы, использующей условный оператор.
3. Построение сложных условий. Демонстрационные примеры.
4. Различные способы записи ветвлений. Демонстрационные примеры.
5. Задания для самостоятельного выполнения.
6. **Назначение и описание условного оператора.**

Условный оператор **if** позволяет изменять порядок выполнения операторов в зависимости от некоторого условия, представляющего собой логическое выражение типа Boolean. Довольно часто в программировании встречаются ситуации, когда необходимо бывает проверить некоторое условие, предъявляемое к переменным. Если значения переменных удовлетворяют этому условию, выполняется одна группа операторов. Если же значения переменных условию не удовлетворяет, то выполняется другая группа операторов или не выполняется ничего. Именно для реализации таких ситуаций в программах используется ***условный оператор***. Он очень похож на условное предложение в разговорной речи:

**ЕСЛИ** <верно условие>, **ТО** <выполни 1-ю группу команд> **ИНАЧЕ** <выполни 2-ю группу команд>.

 В языке Pascal оператор имеет вид:

 **if** <верно условие>, **then** <выполни 1-ю группу команд> **else** <выполни 2-ю группу команд>.

Проверяемое словие имеет только два значения: ИСТИНА или ЛОЖЬ. Если условие, предъявляемое к переменной, оказывается истинным, то выполняется указанная 1-я группа операторов, при этом исполнение группы операторов 2 пропускается. Если условие, предъявляемое к переменным не выполняется (значение условия – ЛОЖЬ), то выполняются операторы группы 2, или управление сразу же передается следующей строке, если ветвь ELSE отсутствует. В качестве операторов, присутствующих в группах команд 1 и 2 условного оператора могут быть любые операторы Pascal .

В качестве условия, предъявляемого к переменным, обычно используют выражения, связанные операциями отношения или логическими операциями.

1. **Особенности структурного построения программы, использующей условный оператор.**

Проверяемое условие в условном операторе это выражение, содержащее одну или несколько логических операций. Результатом выполнения таких логических операций является логическое значение ***истина*** или ***ложь***. Если проверяемое условие истинно, то есть оно выполняется, значение условия равно единице. При ложных значениях, когда условие не выполняется, его значение равно нулю.

Таблица 1. Результаты выполнения логических операций.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |
|  | **Операция** | **Действие** | **Выражение** | **А** | **В** | **Результат** |
|  | NOT | Логическое | NOT A | ИСТИНА |   | ЛОЖЬ |
|  |  | отрицание |  | ЛОЖЬ |  | ИСТИНА |
|  |  |  |  |  |  |  |
|  | AND | Логическое И | A AND B | ЛОЖЬ | ЛОЖЬ | ЛОЖЬ |
|  |  |  |  | ЛОЖЬ | ИСТИНА | ЛОЖЬ |
|  |  |  |  | ИСТИНА | ЛОЖЬ | ЛОЖЬ |
|  |  |  |  | ИСТИНА | ИСТИНА | ИСТИНА |
|  |  |  |  |  |  |  |
|  | OR | Логическое | A OR B | ЛОЖЬ | ЛОЖЬ | ЛОЖЬ |
|  |  | ИЛИ |  | ЛОЖЬ | ИСТИНА | ИСТИНА |
|  |  |  |  | ИСТИНА | ЛОЖЬ | ИСТИНА |
|  |   |   |   | ИСТИНА | ИСТИНА | ИСТИНА |

1. **Построение сложных условий. Демонстрационные примеры.**

 Для построения сложных условий в условных операторах применяются логические операции, объединяющие [выражения отношения](http://informatics-lesson.ru/pascal/expressions-operations.php).

Сформулируем правила для построения сложных условий:

1. Сложное условие строится с учетом приоритетов логических операций и скобок. В порядке убывания приоритета, логические операции расположены так: отрицание, конъюнкция, дизъюнкция. Кроме того, на порядок выполнения операций влияют скобки, которые можно использовать в логических формулах.
2. Операции отношения в языке Паскаль (предсавлены знаками >, >=, <, <=, =, <>) имеют низший приоритет, поэтому в сложном условии они берутся в скобки. .Операции отношения предназначены для сравнения между собой чисел и числовых выражений.
3. Если в сложном условии используются операции равного приоритета, то они выполняются последовательно слева направо.

Примеры записей проверяемых условий на языке Pascak

|  |  |  |
| --- | --- | --- |
|  |  Выражение | Запись на языке Pfscal |
| **1.** | (x ≥ 0) или (y2 ≠ 4) | **(x>=0) or (sqr(y)<>4)** |
| **2.** | xy ≠ 0) и (y > x) | **(x\*y<>0) AND (y>x)** |
| **3.** | не (xy > 0) или (y > x) | **not (x\*y>0) or (y>x)** |
| **4.** | (x > 2) и (y > 3) | **(x > 2) and (y > 3)** |
| **5.** | (x > 1) или (y > -2) | (x > 1) or (y > -2) |
| **6.** | Неверно, что x > 2 | **Not(x>2)** или $\overline{(x-2)}$  |
| **7.** | 10 < x ≤ 20 | **(x > 10) and (x <= 20)** |
| **8.** | Только одно из чисел А и В чётное | **((A mod 2 = 0) and (B mod 2 <> 0)) or** **((A mod 2 <> 0) and (B mod 2= 0))** |
| **9.** | Каждое из чисел А, В, С кратно трём  | **(A mod 3 = 0) and (B mod 3 <> 0) and** **(C mod 3 = 0)**  |
| **10.** | Только одно из чисел А, В, С отрицательное  | **(A < 0) and (B>= 0) and (C >= 0) or** **(A>= 0) and (B < 0) and (C >= 0) or** **(A>= 0) and (B>= 0) and (C < 0)** |
| **11.** | Целое А не кратно трём и оканчивается нулём | **(A mod 3 <> 0) and (A mod 10 = 0)**  |
| **12.** | На числовой оси выделен диапазон значений от 1 до 10, за исключением 5. Составить условие для этого диапазона. | **((x>=3 and (x<5)) or ((x > 5) and (x <= 10))** |

**Пример 1.** Даны три отрезка. Можно ли из них построить треугольник?

Решение. Для любого треугольника сумма его двух сторон  должна быть больше третьей стороны (неравенство треугольника). Если *a, b, c*– стороны предполагаемого треугольника, то нужно сравнить длину каждого отрезка с суммой двух других. Если хотя бы в одном случае отрезок окажется больше суммы двух других, то треугольник из таких отрезков не построить.

Для решения этой задачи используется оператор **if—then—else** со сложным условием.

if (a < b+c) and (b < a+c) and (c < a+b) then

**program** primer1; {Полный условный оператор}

**var** a, b, c: integer;

**begin**

write (' Введите длины отрезков: ');

 readln (a,b,c); {Метки переменных, которым присваиваются }

 {числовые значения, можно указывать списком,}

 {разделённых запятыми}

 **if** (a < b+c) **and** (b < a+c) **and** (c < a+b) **then**

writeln ('Можно построить треугольник.')

 **else**

writeln ('Нельзя построить треугольник.');

**end**.

Пример 2. Составить алгоритм и написать программу для вычисления значения функции f(x): $f\left(x\right)=\left\{\begin{array}{c}x^{2}, если x<0, \\\left|x\right|, в противном случае \end{array}\right.$

Значение **х** вводится с клавиатуры ЭВМ.

**program** primer2; {Полный условный оператор}

**var** x,f: real;

**begin**

write (' Введите x – значение аргумента ');

 readln (x);

 **if** x < 0 **then** writeln ('f= ', sqr(x))

 **else** writeln ('f= ', abs(x))

**end**.

1. **Различные способы записи ветвлений. Демонстрационные примеры.**

В качестве оператора после then и else можно использовать условный оператор. Например, возможна следующая конструкция:

**if** <условие1> **then**

**if** <условие2> **then** <оператор1>

**else** <оператор2>

При использовании таких сложных конструкций (их ещё называют вложенными ветвлениями) следует иметь в виду, что else всегда относится к ближайшему оператору **if**.

Пример. Воспользуемся вложенным ветвлением для записи на языке Паскаль алгоритма решения линейного уравнения ax + b = 0.

Program LinUravn1;

var a, b, x:real;

begin

writeln (‘Решение линейного уравнения’);

write (‘Введите коэффициенты a, b’);

readln (a,b);

if a <> 0 then

 begin

 x:= -b/a;

writeln (‘Корень уравнения х= ’, х);

 end

else if b<> 0 then writeln (‘Корней нет’);

 else writeln (‘x – любое число’);

end.

Как правило, для решения одной и той же задачи можно предложить несколько алгоритмов. Убедимся в этом, записав программу решения линейного уравнения, не прибегая к вложенным ветвлениям.

Program LinUravn2;

var a, b, x:real;

begin

writeln (‘Решение линейного уравнения’);

write (‘Введите коэффициенты a, b’);

readln (a,b);

if a <> 0 then

 begin

 x:= -b/a;

writeln (‘Корень уравнения х= ’, х);

 end;

if (a = 0) and (b<> 0 then writeln (‘Корней нет’);

if (a = 0) and (b = 0) then writeln (‘x – любое число’);

end.

Возможно, второй вариант программы покажется более наглядным. Но и у первого варианта есть свои преимущества: в нём делается меньше проверок.

**Пример.**На вход программе поступают три целых числа. Выведите наибольшее из них (программа должна вывести ровно одно число).

**Решение.** Для решения используем вложенные условные операторы.



**Демонстрационные примеры программ, имеющих ветвящуюся алгоритмическую структуру.**

Задание 1

Дано двузначное число. Определить, какая из его цифр больше, первая или вторая. Составить алгоритм (блок-схему) и написать программу.

{Дано двузначное натуральное число. Определить,}

{какая из цифр больше, первая или вторая}

**program** ЦифрыЧисла2;

**var** x,a,b:integer; {Обозначения введённого числа Х, числа десятков и единиц}

 **begin**

write('х= ');readln(x);

 a:=x **div** 10;writeln('a= ',a);

 b:=(x-a\*10);writeln('b= ',b);

 **if** a>b **then** write('a>b')

 **else** write('b>a');

 **end**.

1. **Упражнения для самостоятельного выполнения.**

**Упражнение 5.1.** Примеры записей проверяемых условий на языке Pascak

|  |  |  |
| --- | --- | --- |
|  |  Выражение | Запись на языке Pfscal |
| **1.** | (x ≥ 0) и (y2 ≠ 4) |  |
| **2.** | xy ≠ 0) и (y > x) |  |
| **3.** | не (xy > 0) или (y > x) |  |
| **4.** | (x > 2) и (y > 3) |  |
| **5.** | (x > 1) или (y > -2) |  |
| **6.** | Неверно, что x > 2 |  |
| **7.** | 10 < x ≤ 20 |  |
| **8.** | Только одно из чисел А и В чётное |  |
| **9.** | Каждое из чисел А, В, С кратно трём  |  |
| **10.** | Только одно из чисел А, В, С отрицательное  |  |
| **11.** | Целое А не кратно трём и оканчивается нулём |  |

**Упражнение 5.2.** Известна длина окружности. Найти площадь круга, ограниченного этой окружностью. Создать блок-схему алгоритма решения и написать программу на языке Pascal.

**Упражнение 5.3. Дано целое число. Определить оканчивается ли оно цифрой семь.** Создать блок-схему алгоритма решения и написать программу на языке Pascal.

**Упражнение 5.4.** Определить, имеется ли среди чисел$a, b, c$ хотя бы одна пара взаимно противоположных чисел. Создать блок-схему алгоритма решения и написать программу на языке Pascal.