**МИНИСТЕРСТВО СЕЛЬСКОГО ХОЗЯЙСТВА РОССИЙСКОЙ ФЕДЕРАЦИИ**

**ФГБОУ ВО «ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ АГРАРНЫЙ УНИВЕРСИТЕТ ИМЕНИ ИМПЕРАТОРА ПЕТРА I»**

**Кафедра информационного обеспечения и моделирования агроэкономических систем**

Курсовая работа

**по дисциплине**

**«Информационные системы и технологии» на тему:**

**Современные информационные системы в финансовой деятельности**

Выполнил студент ПИ-1У

Меснянкин И.А Шифр – 7-221023

Руководитель к.э.н., доц.

Горюхина Е.Ю.

**Воронеж**

Введение

Язык программирования Delphi — один из самых популярных инструментов для разработки программных продуктов. Он появился в середине 90-х годов и с тех пор активно используется разработчиками по всему миру. Delphi сочетает в себе удобный визуальный интерфейс для проектирования приложений и мощный язык программирования Object Pascal, что делает его отличным выбором как для начинающих программистов, так и для опытных специалистов.

Одной из ключевых особенностей Delphi является наличие мощных управляющих структур, которые позволяют реализовывать сложные алгоритмы и обрабатывать данные на высоком уровне. Управляющие структуры, такие как условия и циклы, играют важнейшую роль в программировании. Они обеспечивают возможность принятия решений в процессе выполнения программы, а также дают возможность многократного выполнения определенного кода. Именно благодаря этим механизмам программа может адаптироваться к различным сценариям работы, эффективно решать задачи, связанные с обработкой больших объемов данных, и взаимодействовать с пользователем.

Сложные условия позволяют задавать множество ветвей выполнения программы, что необходимо для реализации гибкой логики. Важно понимать, как использовать логические операторы для комбинирования условий, а также уметь применять такие конструкции, как if..then..else и case..of. Эти конструкции позволяют программе принимать решения на основе заранее заданных правил или пользовательского ввода.

Циклы, в свою очередь, обеспечивают повторное выполнение одного и того же участка кода. Это особенно важно, когда нужно обрабатывать массивы данных, выполнять вычисления или строить итеративные алгоритмы. В Delphi реализованы все основные типы циклов: for, while и repeat, каждый из которых имеет свои особенности и область применения.

Целью данной работы является изучение и анализ управляющих структур языка Delphi, а именно сложных условий и циклов. В работе будут рассмотрены синтаксис, примеры использования и практические рекомендации, которые помогут лучше понять и применять эти инструменты в реальных проектах. А также будет выполнена практическая работа по разработке информационной системы для складского учёта.

Теоретическая часть

# 4 Управляющие конструкции if, case. Циклы.

**Управляющие структуры языка Delphi**

На практике редко встречаются задачи, алгоритм решения которых является линейным. Часто оказывается, что алгоритм решения даже элементарной задачи не является линейным. Например, пусть надо вычислить по формуле ток в электрической цепи. Если предположить, что пользователь всегда будет вводить верные данные, то алгоритм решения этой задачи действительно является линейным. Однако полагаться на то, что пользователь будет вести себя так, как надо программе, не следует. Формула расчета предполагает, что величина сопротивления не равна нулю. А что будет, если пользователь введет 0? Ответ простой: возникнет ошибка "Деление на ноль", и программа аварийно завершит работу. Можно, конечно, возложить ответственность за это на пользователя, но лучше внести изменения в алгоритм решения, чтобы расчет выполнялся только в том случае, если введены верные данные.

Точки алгоритма, в которых выполняется выбор дальнейшего хода программы, называются точками выбора. Выбор очередного шага решения задачи осуществляется в зависимости от выполнения некоторого условия.

**Условие**

В повседневной жизни условие обычно формулируется в виде вопроса, на который можно ответить **Да** или **Нет.**Например:

* Величина сопротивления равна нулю?
* Ответ правильный?
* Сумма покупки больше 300 рублей?

В программе *условие —*это выражение логического типа (Boolean), которое может принимать одно из двух значений: True (истина) или False (ложь).

Простое условие состоит из двух операндов и оператора сравнения. В общем виде условие записывается следующим образом:

Оn1 Оператор On2

где:

* On1 и Оп2 — операнды условия, в качестве которых может выступать переменная, константа, функция или выражение;
* *Оператор —*оператор сравнения.

В языке Delphi есть шесть операторов сравнения, которые приведены в табл. 2.1.

***Таблица 2.1.****Операторы сравнения*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  | **Оператор** | **Описание** | **Результат сравнения** |  |
|  | > | Больше | True, если первый операнд больше второго, иначе False |  |
|  | < | Меньше | True, если первый операнд меньше второго, иначе False |  |
|  | = | Равно | True, если первый операнд равен второму, иначе  False |  |
|  |  |  |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  | **Оператор** | **Описание** | **Результат сравнения** |  |
|  | <> | Не равно | True, если первый операнд не равен второму, иначе False |  |
|  | >= | Больше или равно | True, если первый операнд больше или равен второму, иначе False |  |
|  | <= | Меньше или равно | True, если первый операнд меньше или равен второму, иначе False |  |
|  |  |  |  |  |

Ниже приведены примеры условий:

Summa < 1000 Score >= HBound Sim = Chr(13)

В первом примере операндами условия является переменная и константа. Значение этого условия зависит от значения переменной Summa. Условие будет верным и, следовательно, иметь значение True, если значение переменной Summa меньше, чем 1000. Если значение переменной Summa больше или равно юоо, то значение этого условия будет False.

Во втором примере в качестве операндов используются переменные. Значение этого условия будет True, если значение переменной Score больше или равно значению переменной HBound.

В третьем примере в качестве второго операнда используется функция. Значение этого условия будет True, если в переменной Sim находится символьный код клавиши <Enter>, равный 13.

При записи условий следует обратить особое внимание на то, что операнды условия должны быть одного типа или, если тип операндов разный, то тип одного из операндов может быть приведен к типу другого операнда. Например, если переменная Key объявлена как integer, то условие

Key = Chr(13)

синтаксически неверное, т. к. значение возвращаемое функцией Chr имеет тип char (символьный).

Во время трансляции программы при обнаружении неверного условия компилятор выводит сообщение: incompatible types (несовместимые типы).

Из простых условий при помощи логических операторов: and — "логическое И", or -- "логическое ИЛИ" и not - "отрицание" можно строить сложные условия.

В общем виде сложное условие записывается следующим образом:

*условие1 оператор условие2*

где:

* *условие1*и условие2 — простые условия (выражения логического типа);
* *оператор —***оператор**and или or.

Например:

(ch >= '0') **and**(ch <= '9')

(day = 7) or (day = 6)

(Forml.Editl.Text <> ' ' ) or (Forml.Edit2.Text <> '' )

Forml.CheckBoxl.Checked **and**(Forml.Editl.Text <> '' )

Результат выполнения логических операторов and, or и not представлен в табл. 2.2.

***Таблица 2.2.****Выполнение логических операций*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |
|  | **Op1** | **Op2** | **Opt and Op2** | **Op1 or Op2** | **not Op1** |  |
|  | False | False | False | False | True |  |
|  | False | True | False | True | True |  |
|  | True | False | False | True | False |  |
|  | True | True | True | True | False |  |
|  |  |  |  |  |  |  |

При записи сложных условий важно учитывать то, что логические операторы имеют более высокий приоритет, чем операторы сравнения, и поэтому простые условия следует заключать в скобки.

Например, пусть условие предоставления скидки сформулировано следующим образом: "Скидка предоставляется, если сумма покупки превышает 100 руб. и день покупки — воскресенье", Если день недели обозначен как переменная Day целого типа, и равенство ее значения семи соответствует воскресенью, то условие предоставления скидки можно записать:

(Summa > 100) **and**(Day = 7)

Если условие предоставления скидки дополнить тем, что скидка предоставляется в любой день, если сумма покупки превышает 500 руб., то условие можно записать:

((Summa > 100) **and**(Day =7)) **or**(Summa > 500)

**Циклы**

Алгоритмы решения многих задач являются циклическими, т. е. для достижения результата определенная последовательность действий должна быть выполнена несколько раз.

Например, программа контроля знаний выводит вопрос, принимает ответ, добавляет оценку за ответ к сумме баллов, затем повторяет это действие еще и еще раз, и так до тех пор, пока испытуемый не ответит на все вопросы.

Другой пример. Для того чтобы найти фамилию человека в списке, надо проверить первую фамилию списка, затем вторую, третью и т. д. до тех пор, пока не будет найдена нужная фамилия или не будет достигнут конец списка.

Алгоритм, в котором есть последовательность операций (группа инструкций), которая должна быть выполнена несколько раз, называется *циклическим,*а сама последовательность операций именуется *циклом.*

В программе цикл может быть реализован при помощи инструкций for,

while и repeat.