

Лабораторная работа №3

Тема работы: Простые типы данных. Ввод и вывод простых типов данных.

Задание:

Часть 1.

Создать приложение, реализующее следующее множество математических операций:

1. Сложение
2. Вычитание
3. Умножение
4. Деление
5. Взятие остатка
6. Логическое и побитовое умножение (AND)
7. Логическое и побитовое сложение (OR)
8. Логическое и побитовое сложение по модулю 2 (XOR)
9. Логическое и побитовое отрицание (NOT)
10. Больше
11. Меньше
12. Больше либо равно
13. Меньше либо равно
14. Равно

Для целочисленных, вещественных и логических данных.

Приложение должно проверять введенные аргументы на корректность и соответствие типу данных. Константе «истина» соответствует текст «true» (без кавычек), константе «ложно» — текст «false» (без кавычек).

В случае некорректного ввода или несоответствия типа данных пользователь должен получить соответствующее сообщение, с указанием конкретной ошибки и аргумента, содержащего ошибку.

Результат вычисления должен вместе с аргументами в виде полного выражения помещаться в конец компонента «спискок». Опционально — предусмотреть функцию очистки истории вычислений.

Часть 2.

Дополнить приложение математическими функциями согласно варианту.

Выполнение:

Для начала необходимо создать пользовательский интерфейс приложения, по аналогии с рисунком. После размещения всех необходимых компонентов (каких именно?) в дизайнере форм, следует определить основные рабочие процедуры. Все вычисления мы будем проводить, пользуясь стандартными математическими функциями библиотеки VCL(LCL).

Сперва нам нужно получить от пользователя аргументы. Пользователь заносит их в поля ввода, содержимое которых хранится в системе в виде строки символов. Соответственно, эти строки необходимо преобразовать в переменные соответствующих типов, например, целые. Для этого в стандартной библиотеке, в модуле SysUtils есть функции StrToInt для преобразования строки в целое число, StrToFloat — в вещественное и StrToBool — в логическое. Для обратного преобразования служат функции IntToStr, FloatToStr и BoolToStr.

Рассмотрим пример (считается, что поля ввода для аргументов не переименовывались, и имеют имена по-умолчанию Edit1 и Edit2):

procedure Button1Click(Sender: TObject);

var

A, B, Res: Integer;

begin

A := StrToInt(Edit1.Text);

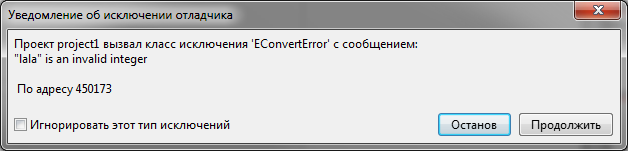
B := StrToInt(Edit2.Text);

Res := A+B;

ShowMessage('Результат: '+IntToStr(Res));

end;

Однако, в случае невозможности преобразования эти функции выбрасывают исключительную ситуацию (ошибку), которая сопровождается сообщением следующего вида:



В случае запуска приложения без отладчика (или продолжения приложения с) будет выдано сообщение более привычного вида с предложением закрыть программу.

Чтобы избежать подобных ситуаций, в целом пугающих пользователей, необходимо каким-то образом избавиться от этих сообщений и заменить их на более понятные. Для этого существуют 2 подхода: первый, это обработать исключительную ситуацию (об этом мы поговорим позднее), второй, это не допустить ее появления. В данном случае можно легко реализовать 2-й подход с использованием функций с приставкой Try. Эти функции имеют 2 параметра — строку для преобразования и результат преобразования и возвращают логическое значение, удалось преобразование или же нет. Обратите внимание, что обратное преобразование (число в строку) всегда возможно, и функции преобразования чисел в строку не имеют таких «безопасных» версий. Рассмотрим пример:

procedure Button1Click(Sender: TObject);

var

A, B, Res: Integer;

begin

if TryStrToInt(Edit1.Text, A) then

begin

. . .

end

else

Application.MessageBox(PChar('Мне кажется, что '+Edit1.Text+ ' не выглядит похожим на число'), 'Ошибка ввода', mb\_iconerror);

end;

В примере показана обработка неправильного ввода для одного из аргументов. Помните, что для успешного завершения операции требуется, чтобы одновременно ОБА аргумента были введены корректно. Также в примере пользователю показывается сообщение о неправильно введенном параметре (для его работы требуется подключить модуль Windows, добавив его в секцию uses модуля формы, если его там еще нет).

Для хранения параметра в примере использован целочисленный тип Integer. Это не единственный предопределенный целочисленный тип в Object Pascal, другие целочисленные типы и их диапазон значений можно посмотреть в таблице:

|  |  |  |
| --- | --- | --- |
| Тип | Диапазон | Размер |
| **Integer** | -2147483648..2147483647 | Знаковый 4 байт (32 бит) |
| **Cardinal** | 0..4294967295 | Беззнаковый 4 байт (32 бит) |
| **ShortInt** | -128..127 | Знаковый 1 байт (8 бит) |
| **SmallInt** | -32768..32767 | Знаковый 2 байт (16 бит) |
| **LongInt** | -2147483648..2147483647 | Знаковый 4 байт (32 бит) |
| **Int64** | -263..263-1 | Знаковый 8 байт (64 бит) |
| **Byte** | 0..255 | Беззнаковый 1 байт (8 бит) |
| **Word** | 0..65535 | Беззнаковый 2 байт (16 бит) |
| **LongWord** или **DWORD** | 0..4294967295 | Беззнаковый 4 байт (32 бит) |
| **UInt64** | 0..264-1 | Беззнаковый 8 байт (64 бит) |

Чаще всего используются типы Integer и LongWord. Размеры и диапазон типов Integer и Cardinal даны для 32 битного режима. В 16 битном режиме их размер 16 бит (2 байта) в 64 битном в большинстве случаев такой же, как и в 32 битном, для обеспечения совместимости.

Помимо целых типов, существуют вещественные — которые хранят числа с плавающей запятой (т.е. нецелые).

|  |  |  |  |
| --- | --- | --- | --- |
| Тип | Диапазон | Кол-во разрядов | Размер (байт) |
| **Real48** | 2.9 × 10-39 .. 1.7 × 1038 | 11-12 | 6 |
| **Single** | 1.5 × 10-45 .. 3.4 × 1038 | 7-8 | 4 |
| **Double** | 5.0 × 10-324 .. 1.7 × 10308 | 15-16 | 8 |
| **Extended** | 3.6 × 10-4951 .. 1.1 × 104932 | 10-20 | 10 |
| **Comp** | -263+1 .. 263-1 | 10-20 | 8 |
| **Currency** | -922337203685477.5808.. 922337203685477.5807 | 10-20 | 8 |

Тип Real48 оставлен для совместимости с программами, написанными на языке Turbo Pascal для процессоров не имеющих блока вещественных операций, и сегодня абсолютно бесполезен. Чаще всего используются типы Single (вещественное одинарной точности), Double (вещественное двойной точности) и, реже, Extended (вещественное машинной точности). В данной работе рекомендуется использовать тип Double, т.к. тип Single из-за особенностей обработки вещественных чисел в ЭВМ может давать отличные от калькулятора значения уже в 7-8 знаке после запятой.

Логический тип имеет всего два значения, это «истина» («true») и «ложь» («false»). Основным логическим типом в языке Object Pascal является тип Boolean. Любое логическое выражение имеет тип Boolean. Для проверки логических выражений используется условный оператор if:

If выражение then

Оператор

[else

Оператор]

Ветка else язвляется необязательной. В случае ее наличия точка с запятой после оператора истинной ветки и соответственно перед else НЕ СТАВИТСЯ. Это единственный случай в языке Object Pascal, где нельзя ставить точку с запятой после оператора.

Пример использования логического выражения:

procedure Button100500Click(Sender: TObject);

var

A, B: Integer;

begin

A := StrToInt(Edit1.Text);

B := StrToInt(Edit2.Text);

if A = B then

ShowMessage(IntToStr(A) + ' равно ' + IntToStr(B))

else

ShowMessage(IntToStr(A) + ' не равно ' + IntToStr(B));

end;

1 вариант

1. является ли число степенью 2
2. sin
3. tan
4. e^x

2 вариант

1. является ли число степенью 3
2. cos
3. ctan
4. x^y

3 вариант

1. является ли число степенью 4
2. arctan
3. arcsin
4. корень степени y из x

4 вариант

1. является ли число степенью 5
2. arcctan
3. arccos
4. ln x