СИСТЕМА УПРАВЛЕНИЯ БАЗАМИ ДАННЫХ

ЛИНТЕР®

ЛИНТЕР БАСТИОН

ЛИНТЕР СТАНДАРТ

# Библиотеки специальных типов данных

НАУЧНО-ПРОИЗВОДСТВЕННОЕ ПРЕДПРИЯТИЕ

РЕЛЭКС

## Товарные знаки

РЕЛЭКС™, ЛИНТЕР® являются товарными знаками, принадлежащими АО НПП «Реляционные экспертные системы» (далее по тексту – компания РЕЛЭКС). Прочие названия и обозначения продуктов в документе являются товарными знаками их производителей, продавцов или разработчиков.

## Интеллектуальная собственность



## О документе

Материал, содержащийся в данном документе, прошел доскональную проверку, но компания РЕЛЭКС не гарантирует, что документ не содержит ошибок и пропусков, поэтому оставляет за собой право в любое время вносить в документ исправления и изменения, пересматривать и обновлять содержащуюся в нем информацию.

## Контактные данные

394006, Россия, г. Воронеж, ул. Бахметьева, 2Б.

Тел./факс: (473) 2-711-711, 2-778-333.

e-mail: market@relex.ru.

## Техническая поддержка

С целью повышения качества программного продукта ЛИНТЕР и предоставляемых услуг в компании РЕЛЭКС действует автоматизированная система учёта и обработки пользовательских рекламаций. Обо всех обнаруженных недостатках и ошибках в программном продукте и/или документации на него просим сообщать нам в раздел Поддержка на сайте ЛИНТЕР.

# Содержание

# Предисловие

## Назначение документа

Документ содержит описание библиотек специальных типов данных для работы с числами с фиксированной точкой, типом «дата-время» и длинными целыми числами, реализованными в СУБД ЛИНТЕР, но не поддерживаемыми компиляторами языка C/C++. Библиотеки совместимы со всеми программными платформами, на которых функционирует СУБД ЛИНТЕР.

Документ предназначен для СУБД ЛИНТЕР СТАНДАРТ 6.0 сборка 20.1, далее по тексту СУБД ЛИНТЕР.

## Для кого предназначен документ

Документ предназначен для программистов, разрабатывающих приложения на языке программирования C/C++ под управление СУБД ЛИНТЕР.

## Необходимые предварительные знания

Для работы с утилитой необходимо:

- знать язык программирования C/C++;
- уметь работать в соответствующей операционной системе на уровне простого пользователя.

## Дополнительные документы

- СУБД ЛИНТЕР. Архитектура СУБД
- СУБД ЛИНТЕР. Справочник по SQL
- СУБД ЛИНТЕР. Справочник кодов завершения

# Библиотека Decimals

## Назначение

Библиотека DECIMALS предназначена для работы с вещественными числами с фиксированной точкой (тип данных `DECIMAL` СУБД ЛИНТЕР). Необходимость введения этого типа данных вызвана тем, что стандартные типы данных языка C/C++ не обеспечивают высокой точности, необходимой, например, для бухгалтерских расчетов.

Набор функций библиотеки обеспечивает выполнение:

1) арифметических операций;
2) сравнение чисел между собой;
3) преобразование из строкового представления во внутренний формат СУБД ЛИНТЕР;
4) преобразование из внутреннего формата СУБД ЛИНТЕР в строковое представление;
5) преобразование к другим типам данных.

## Условия применения

Библиотека может использоваться только в программах на языке C/C++.

Модуль языка C/C++, в котором предполагается использовать функции библиотеки, должен включать заголовочный файл `decimals.h`.

В проект приложения должна быть добавлена библиотека `decimals.a`.

**Примечание**
Файлы `decimals.h` и `decimals.a` входят в состав поставки СУБД ЛИНТЕР.

## Характеристики библиотеки

Вещественные числа с фиксированной точкой хранятся в базе данных в специальном внутреннем формате, который обеспечивает:

1) максимальное количество значащих десятичных цифр – 30 (20 в целой части числа);
2) максимальную точность (число цифр после десятичной точки) – 10.

Число в этом формате занимает 16 байт. Первый байт числа содержит специальную информацию, которую в терминах языка C/C++ можно представить следующим образом:

```
typedef enum {
   DECZERO,     /* 0 *//* Значение нулевое */
   DECNEGATIV, /* 1 *//* Значение отрицательное */
   DECPOSITIV, /* 2 *//* Значение положительное */
   DECNEGOVER, /* 3 *//* Переполнение отрицательного */
   DECPOSOVER, /* 4 *//* Переполнение положительного */
   DECERROR    /* 5 *//* Ошибка */
 } DECSTATUS;
```

Таким образом, первый байт внутреннего представления числа хранит информацию о знаке числа либо об ошибке (в частности, о переполнении).

Все операции с вещественными числами с фиксированной точкой ведутся с количеством значащих цифр 20 до запятой и 10 – после запятой. Явное указание значений этих параметров влияет только на символьное представление числа в операциях преобразования.

При преобразованиях целых чисел существует ограничение: число -2147483648, допустимое в языке программирования Си, в библиотеке недопустимо.

При преобразовании вещественных чисел следует помнить, что преобразование это не точное, а выполняется с точностью вещественного числа.

# Библиотечные функции

При описании функций библиотеки используются обозначения, определенные в заголовочном файле `decimals.h`.

# Арифметические функции

## Сложение

### Прототип

```
void ADDDECIMAL (
   DECIMAL D1,   /* первое слагаемое */
   DECIMAL D2,   /* второе слагаемое */
   DECIMAL S);   /* сумма */
```

### Описание

Процедура `ADDDECIMAL` складывает два вещественных числа с фиксированной точкой, представленных в переменных `D1` и `D2` типа `DECIMAL`.

### Возвращаемое значение

Сумма значений `D1` и `D2` помещается в переменную `S` типа `DECIMAL`, статус результата операции фиксируется в первом байте значения переменной `S`.

## Вычитание

### Прототип

```
void SUBDECIMAL (
   DECIMAL D1,   /* уменьшаемое */
   DECIMAL D2,   /* вычитаемое */
   DECIMAL D);   /* разность */
```

### Описание

Процедура `SUBDECIMAL` вычитает из одного вещественного числа с фиксированной точкой, представленного в переменной `D1` типа `DECIMAL`, другое вещественное число с фиксированной точкой, представленное в переменной `D1` типа `DECIMAL`.

 

### Возвращаемое значение

Разность значений D1 и D2 помещается в переменную D типа DECIMAL, статус результата операции фиксируется в первом байте значения переменной D.

## Сравнение двух чисел

### Прототип

```
INT CMPDECIMAL (
   DECIMAL D1,  /* первое число */
   DECIMAL D2); /* второе число */
```

### Описание

Функция CMPDECIMAL выполняет арифметическое сравнение двух вещественных чисел с фиксированной точкой, представленных в переменных D1 и D2 типа DECIMAL.

### Возвращаемое значение

Результат сравнения (значение типа INT):

1) +1, если D1 > D2;
2) 0, если D1 = D2;
3) −1, если D1 < D2.

### Пример

```
printf('d1 ');
if ((c =CMPDECIMAL(d1,d2)) == 0)
   printf('=');
else
   printf('%c' ,(c ==1)? '>':'<');
printf('d2');
```

## Умножение

### Прототип

```
void MULDECIMAL (
   DECIMAL D1,   /* первый сомножитель */
   DECIMAL D2,   /* второй сомножитель */
   DECIMAL M);   /* произведение */
```

### Описание

Процедура MULDECIMAL перемножает два вещественных числа с фиксированной точкой, представленных в переменных D1 и D2 типа DECIMAL.

### Возвращаемое значение

Произведение значений D1 и D2 помещается в переменную M типа DECIMAL, статус результата операции фиксируется в первом байте значения переменной M.

## Деление

### Прототип

```
void DIVDECIMAL (
   DECIMAL D1,    /* делимое */
   DECIMAL D2,    /* делитель */
   DECIMAL D);    /* частное */
```

### Описание

Процедура DIVDECIMAL выполняет целочисленное деление двух вещественных чисел с фиксированной точкой, представленных в переменных D1 и D2 типа DECIMAL.

### Возвращаемое значение

Целая часть от деления значений D1 и D2 помещается в переменную D типа DECIMAL, статус результата операции фиксируется в первом байте значения переменной D.

# Функции преобразования

Все функции преобразования возвращают целое число типа INT – флаг успешности преобразования:

1) 1 – при успешном завершении преобразования;

2) 0 – при ошибках преобразования.

## Преобразование из символьного представления во внутреннее

### Прототип

```
INT STRTODEC (
  CHAR * Ss,    /* символьная строка */
  DECIMAL Od); /* преобразованное число */
```

### Описание

Функция STRTODEC преобразует строку, находящуюся в переменной Ss, в вещественное число с фиксированной точкой.

Разделителем целой и дробной части преобразуемого символьного представления числа может быть точка или запятая.

### Возвращаемое значение

Код завершения (0 или 1). При успешном завершении преобразованное число в переменной Od типа DECIMAL.

### Пример

```
…
STRTODEC ("+3.14" ,pdec);
…
```

 

# Преобразование из внутреннего представления в символьное

## Прототип

```
void DECTOSTR (
   DECIMAL d,    /* преобразуемое число */
   CHAR * Os,    /* символьное представление */
   INT Ln,       /* число значащих цифр */
   INT Lns,      /* точность преобразования */
   INT IsSgn);   /* признак необходимости знака */
```

## Описание

Функция DECTOSTR преобразует внутреннее представление вещественного числа с фиксированной точкой, заданного переменной d, в его символьное представление в соответствии с указанным форматом:

1) Ln – общее число знаков в символьном представлении;
2) Lns – число знаков после десятичной точки в символьном представлении;
3) IsSgn = 1, знак числа обязательно выводится;
4) IsSgn = 0, выводится только знак минус (-), вместо знака плюс (+) будет выведен пробел.

Значащие нули в результат преобразования не включаются.

## Пример

```
…
DECTOSTR(pdec,Str , 30, 2, 1);
/* 30 знаков в числе, 2 после точки, знак обязателен! */
```

# Преобразование из внутреннего представления в полное символьное

## Прототип

```
void DECTOSTRN (
   DECIMAL d,       /* преобразуемое число */
   CHAR *Os,        /* символьное представление */
   INT Ln,          /* число значащих цифр */
   INT Lns,         /* точность преобразования */
   INT IsSgn,       /* признак необходимости знака */
   INT Cleartail); /* признак преобразования значащих нулей */
```

## Описание

Функция DECTOSTRN преобразует внутреннее представление вещественного числа с фиксированной точкой, заданного переменной d, в его символьное представление в соответствии с указанным форматом:

1) Ln – общее число знаков в символьном представлении;
2) Lns – число знаков после десятичной точки в символьном представлении;

3) `IsSgn = 1`, знак числа обязательно выводится;

4) `IsSgn = 0`, выводится только знак минус (-), вместо знака плюс (+) будет выведен пробел;

5) `Cleartail = 1`, значащие нули не вставляются;

6) `Cleartail = 0`, значащие нули вставляются.

# Преобразование целого числа

### Прототип

```
INT LongToDec (
   LONG Arg,      /* исходное целое число */
   DECIMAL Od);   /* результат преобразования */
```

### Описание

Функция `LongToDec` преобразует целое число типа `LONG`, представленное переменной `Arg`, в формат вещественного числа с фиксированной точкой.

### Возвращаемое значение

Вещественное значение в переменной `Od` типа `DECIMAL`.

# Преобразование в целое

### Прототип

```
INT DecToLong (
   DECIMAL Id,    /* исходное вещественное число */
   LONG * Ol,     /* преобразованное целое */
   INT Round);    /* признак округления */
```

### Описание

Функция `DecToLong` преобразует вещественное число с фиксированной точкой, заданное переменной `Id` типа `DECIMAL`, в целое число типа `LONG`.

Если преобразуемое вещественное число имеет целочисленное значение, то параметр `Round` игнорируется;

Если `Round = 0`, выполняется округление до целого значения.

Если вещественное число содержит дробную часть и значение `Round = 1`, то функция возвращает ошибку преобразования.

### Возвращаемое значение

Код завершения (`0` или `1`). При успешном завершении – целая часть вещественного числа (без округления) в переменной `Ol` типа `LONG`.

### Пример

```
...
if ((ok =DecToLong(vdec,&LStr ,1)) == 0)
```

 

```
   {   /* Ошибка преобразования с округлением*/
   printf('\n Ошибка:  переполнение при преобразовании');
      goto lend;
      }  /* Ifthen */
```

…

# Преобразование длинного целого числа

### Прототип

```
INT DLongToDec (
   L_DLONG Arg,  /* исходное целое число */
   DECIMAL Od);  /* результат преобразования */
```

### Описание

Функция DLongToDec преобразует длинное целое число типа L_DLONG, представленное переменной Arg, в формат вещественного числа с фиксированной точкой.

### Возвращаемое значение

Код завершения (0 или 1). При успешном завершении – вещественное значение в переменной Od типа DECIMAL.

# Преобразование в целое

### Прототип

```
INT DecToDLong (
   DECIMAL Id,   /* исходное вещественное число */
   L_DLONG *Ol,  /* преобразованное целое */
   Int Round);   /* признак округления */
```

### Описание

Функция DecToDLong преобразует вещественное число с фиксированной точкой, заданное переменной Id типа DECIMAL, в длинное целое число типа L_DLONG.

Если преобразуемое вещественное число имеет целочисленное значение, то параметр Round игнорируется;

Если Round = 0, выполняется округление до целого значения.

Если вещественное число содержит дробную часть и значение Round = 1, то функция возвращает ошибку преобразования

### Возвращаемое значение

Код завершения (0 или 1). При успешном завершении – целая часть вещественного числа (без округления) в переменной Ol типа L_DLONG.

### Пример

…

```
if ((ok =DecToDLong(vdec,&LStr ,1)) == 0)
   {   /* Ошибка преобразования с округлением*/
   printf('\n Ошибка:  переполнение при преобразовании');
   goto lend;
   } /* Ifthen */
…
```

## Преобразование вещественного числа двойной точности

### Прототип

```
INT DblToDec (
   double Idbl, /* исходное число */
   DECIMAL Od); /* преобразованное число */
```

### Описание

Функция `DblToDec` преобразует вещественное число двойной точности, заданное переменной `Idbl` типа `double`, в вещественное число с фиксированной точкой.

Разделителем целой и дробной части преобразуемого вещественного числа двойной точности может быть точка или запятая.

### Возвращаемое значение

Код завершения (`0` или `1`). При успешном завершении – вещественное число с фиксированной точкой в переменной `Od` типа `DECIMAL`.

### Пример

```
…
double d = 1234567890.123456;
…
DblToDec (d ,decvar);
…
```

## Преобразование в вещественное число двойной точности

### Прототип

```
INT DecToDbl (
   DECIMAL Id,     /* исходное число */
   double * Odbl); /* преобразованное число */
```

### Описание

Функция `DecToDbl` преобразует вещественное число с фиксированной точкой, заданное переменной `Id` типа `DECIMAL`, в вещественное число двойной точности.

### Возвращаемое значение

Код завершения (`0` или `1`). При успешном завершении – вещественное число двойной точности в переменной `Odbl` типа `double`.

 

## Копирование числа

### Прототип

```
void COPYDEC (
  DECIMAL Id,  /* переменная-источник */
  DECIMAL Od); /* переменная-приемник */
```

### Описание

Процедура COPYDEC копирует вещественное число с фиксированной точкой, представленное в переменной Id типа DECIMAL, в переменную Od типа DECIMAL.

## Выделение целой части числа

### Прототип

```
void ENTDECIMAL (
   DECIMAL Id,    /* исходное число */
   DECIMAL Od);   /* целая часть числа */
```

### Описание

Процедура выделяет целую часть вещественного числа с фиксированной точкой, представленного в переменной Id типа DECIMAL, и помещает ее в переменную Od типа DECIMAL.

## Округление до заданной точности

### Прототип

```
void RNDDECIMAL (
   DECIMAL Id,    /* исходное число */
   BYTE Prec,     /* масштаб */
   BYTE Scale);   /* точность */
```

### Описание

Процедура RNDDECIMAL преобразует исходное вещественное число с фиксированной точкой, представленное в переменной Id типа DECIMAL, в округленное число с заданным масштабом и точностью. Аргумент Prec задает общее количество цифр в представлении числа, а Scale – количество цифр после десятичной точки.

## Изменение знака числа

### Прототип

```
void NEGDECIMAL (
   DECIMAL Id,    /* исходное число */
   DECIMAL Od);   /* преобразованное число */
```

### Описание

Процедура NEGDECIMAL изменяет знак вещественного числа с фиксированной точкой, представленного в переменной Id типа DECIMAL, на противоположный и помещает его в переменную Od типа DECIMAL.

# Функции работы со статусом числа

## Получение статуса числа

### Прототип

```
DECSTATUS GETDECSTATUS (DECIMAL D);
```

### Описание

Функция GETDECSTATUS возвращает специальную информацию о состоянии вещественного числа с фиксированной точкой, представленного в переменной D типа DECIMAL (см. подраздел «Характеристики библиотеки»).

## Проверка корректности числа

### Прототип

```
INT OKDECSTATUS (DECIMAL D);
```

### Описание

Функция OKDECSTATUS проверяет текущее состояние вещественного числа с фиксированной точкой, представленного в переменной D типа DECIMAL.

### Возвращаемое значение

1) 1 – правильное представление.
2) 0 – ошибка преобразования или переполнение.

## Установка статуса числа

### Прототип

```
void SETDECSTATUS (
   DECSTATUS St,    /* новый статус */
   DECIMAL D);      /* модифицируемое число */
```

### Описание

Процедура SETDECSTATUS устанавливает новый статус вещественного числа с фиксированной точкой, представленного в переменной D типа DECIMAL. Новое значение статуса числа задается в структуре данных St типа DECSTATUS (см. подраздел «Характеристики библиотеки»).

# Прочие функции

## Присвоить максимальное значение

### Прототип

```
void SetMaxDecimal (DECIMAL D);
```

 

### Описание

Процедура `SetMaxDecimal` присваивает переменной `D` типа `DECIMAL` максимальное значение вещественного числа с фиксированной точкой.

## Присвоить минимальное значение

### Прототип

```
void SetMinDecimal (DECIMAL D);
```

### Описание

Процедура `SetMinDecimal` присваивает переменной `D` типа `DECIMAL` минимальное значение вещественного числа с фиксированной точкой.

# Библиотека Tick

## Назначение

Библиотека TICK предназначена для работы со значениями типа «дата/время» (тип данных DATE СУБД ЛИНТЕР). Необходимость введения подобного типа данных вызвана тем, что среди стандартных типов данных языка C/C++ отсутствует тип данных DATE.

Набор функций библиотеки обеспечивает:

1) различные преобразования даты из внутреннего формата в общепринятые символьные представления и обратно;
2) определение всевозможных характеристик, касающихся работы с датой: число дней в конкретном месяце, високосность заданного года и т.п.;
3) аддитивные операции с данными типа «дата/время».

## Условия применения

Библиотека может использоваться только в программах на языке C/C++.

Модуль языка C/C++, в котором предполагается использовать функции библиотеки, должен включать заголовочный файл tick.h.

В проект приложения должен быть добавлен файл tick.a.

**Примечание**
Файлы tick.h и tick.a входят в состав поставки СУБД ЛИНТЕР.

## Характеристики библиотеки

Данные типа «дата/время» хранят дату, совмещенную со временем. Минимальной единицей деления времени в СУБД ЛИНТЕР является *тик* – 1/100 секунды. Это позволяет достаточно точно отражать время какого-либо события или состояния управляемого объекта.

Основные характеристики данных типа ДАТА:

1) дата хранится в переменной типа DECIMAL и занимает 16 байт;
2) дата хранится в виде числа тиков от начала нашей эры (н.э.);
3) максимальный год, который можно отразить в переменной «дата/время» – 9999 г.

## Функции

При описании функций библиотеки используются обозначения, определенные в заголовочных файлах tick.h и decimals.h.

## Функции преобразования

### Преобразование даты к числу дней и числу тиков

#### Прототип

```
void TICKTODATE (
```

```
   DECIMAL Date,    /* исходная дата */
   LONG * days,     /* число дней от н.э. */
   LONG * ticks);   /* число тиков в дне */
```

### Описание

Процедура TICKTODATE определяет полное количество дней и число тиков неполного дня в исходной дате Date и помещает вычисленные значения в переменные days и ticks соответственно.

### Пример

```
DECIMAL Date;
LONG    days;
LONG    ticks;
...
TICKTODATE (Date ,&days ,&ticks);
...
```

## Преобразование номера месяца в трехбуквенное название месяца

### Прототип

```
void NAMMON (
   INT Nmonth,    /* исходный номер месяца */
   CHAR * month, /* выходное название месяца */
   INT koder);    /* тип кодировки названия месяца */
```

### Описание

Функция NAMMON преобразует номер месяца Nmonth (целое число в диапазоне от 1 до 12) в трехбуквенное название месяца, которое размещает в переменной month. Название месяца представляется в зависимости от заданного типа кодировки koder:

1) 0 – english;
2) 1 – CP1251;
3) 2 – KOI8-R;
4) 3 – CP866.

### Возвращаемое значение

1) 1 – успешное завершение.
2) 0 – ошибка преобразования.

### Пример

```
INT   i;
INT   Nmonth;
CHAR  month[8];
...
```

```
i = NAMMON (Nmonth ,month,0);
...
```

## Расширенное преобразование номера месяца в название месяца

### Прототип

```
void NAMMONex (
   INT Nmonth,    /* исходный номер месяца */
   CHAR * month, /* выходное название месяца */
   INT koder,     /* тип кодировки названия месяца */
   INT Count);    /* размер выходного буфера */
```

### Описание

Функция NAMMONex действует аналогично функции NAMMON, за исключением того, что в выходной буфер копируется до Count символов.

### Пример

```
INT   i;
INT   Nmonth;
CHAR  month[8];
...
NAMMONex (Nmonth ,month,0,5);
...
```

## Преобразование трехбуквенного названия месяца в номер месяца

### Прототип

```
INT MONNAM (
   CHAR * month, /* название месяца */
   INT koder);   /* тип кодировки названия месяца */
```

### Описание

Функция MONNAM преобразует трехбуквенное название месяца Month в его номер (целое число в диапазоне от 1 до 12), который возвращает в качестве значения функции. Для правильного преобразования необходимо указать кодировку koder исходного названия месяца (см. подпункт «Преобразование номера месяца в трехбуквенное название месяца»).

### Возвращаемое значение

1) Номер месяца – успешное завершение.
2) 0 – ошибка преобразования.

### Пример

```
INT   i;
```

 

```
CHAR  Month[4];
...
strcpy(Month ,"MAR");
i = MONNAM (Month,0);
...
```

## Расширенное преобразование названия месяца в номер месяца

### Прототип

```
INT MONNAMex (
   CHAR * month, /* название месяца */
   INT koder,    /* тип кодировки названия месяца */
   INT Count);   /* размер входного буфера */
```

### Описание

Функция MONNAMex действует аналогично с MONNAN, за исключением того, что используется не более Count символов в названии месяца. В случае недостаточности Count для точной идентификации месяца используется первое совпадение.

### Возвращаемое значение

1) Номер месяца – успешное завершение.
2) 0 – ошибка преобразования.

### Пример

```
INT   i;
CHAR  Month[6];
...
strcpy(Month ,"march");
i = MONNAMex (Month,0,5);
...
```

## Преобразование номера дня недели в его трехбуквенное название

### Прототип

```
void NAMDAY (
   INT Ndayweek,   /* номер дня недели */
   CHAR * dayweek, /* выходное название дня недели */
   INT koder);     /* тип кодировки названия дня */
```

### Описание

Функция NAMDAY преобразует номер дня недели Ndayweek (целое число в диапазоне от 1 (понедельник) до 7 (воскресенье)) в трехбуквенное название дня недели, которое размещает в переменной dayweek. Название дня недели представляется в зависимости

от заданного типа кодировки koder (см. подпункт «Преобразование номера месяца в трехбуквенное название месяца»).

### Возвращаемое значение

1) 1 – успешное завершение.
2) 0 – ошибка преобразования.

### Пример

```
INT   i;
INT   Ndayweek;
CHAR  dayweek[4];
...
i = NAMDAY (Ndayweek ,dayweek,0);
```

## Расширенное преобразование номера дня недели в его название

### Прототип

```
void NAMDAYex (
   INT Ndayweek,    /* номер дня недели */
   CHAR * dayweek, /* выходное название дня недели */
   INT koder,       /* тип кодировки названия дня */
   INT Count);      /* размер выходного буфера */
```

### Описание

Функция NAMDAYex действует аналогично NAMDAY, за исключением того, что копируется не более Count символов.

### Пример

```
INT   i;
INT   Ndayweek;
CHAR  dayweek[10];
...
NAMDAYex (Ndayweek ,dayweek,0,5);
```

## Преобразование трехбуквенного названия дня недели в его номер

### Прототип

```
INT DAYNAM (
   CHAR * Dayweek); /* название дня недели */
```

### Описание

Функция DAYNAM преобразует трехбуквенное название дня недели Dayweek в его порядковый номер (целое число в диапазоне от 1 (понедельник) до 7 (воскресенье)), который возвращает в качестве значения функции.

 

### Возвращаемое значение

1) Номер дня недели – успешное завершение.
2) 0 – ошибка преобразования.

### Пример

```
INT   i;
   CHAR  Dayweek[3];
   ...
   strcpy(Dayweek ,"Суб");
   i = DAYNAM (Dayweek);
   ...
```

## Расширенное преобразование названия дня недели в его номер

### Прототип

```
INT DAYNAMex (
   CHAR * Dayweek, /* название дня недели */
   INT Count);     /* размер входного буфера */
```

### Описание

Функция DAYNAMex выполняется аналогично функции DAYNAM, за исключением того, что в названии дня недели используется не более Count символов. В случае недостаточности Count для точной идентификации дня недели используется первое совпадение.

### Возвращаемое значение

1) Номер дня недели – успешное завершение.
2) 0 – ошибка преобразования.

### Пример

```
INT   i;
CHAR  Dayweek[10];
...
strcpy(Dayweek ,"saturday");
i = DAYNAMex (Dayweek,6);
...
```

## Преобразование даты из символьной строки во внутренний формат

### Прототип

```
INT STRTOTICK (
   CHAR * Str,    /* дата в символьном виде */
```

```
   DECIMAL date); /* дата во внутреннем формате */
```

### Описание

Функция STRTOTICK преобразует дату, представленную в виде символьной строки Str формата:

```
"dd.mm.yyyy[:[h]h[:[m]m[:[s]s[.[t]t]]]]"
```

или

```
"dd/mm/yyyy[:[h]h[:[m]m[:[s]s[.[t]t]]]]"
```

во внутренний формат представления даты date-переменную типа DECIMAL.

### Возвращаемое значение

1) 1 – успешное завершение.
2) 0 – ошибка преобразования.

### Пример

```
INT   i;
DECIMAL  date;
CHAR * Str = "15.05.1994:12:3:05.75";
...
i = STRTOTICK (Str ,date);
...
```

## Преобразование даты в символьный вид

### Прототип

```
void TICKTOSTR (
   DECIMAL Date,   /* дата во внутреннем формате */
   INT DateFormat, /* формат вывода даты */
   INT YearFormat, /* формат вывода года */
   CHAR * str,     /* символьный вид даты */
   INT LengthStr); /* длина выходной строки */
```

### Описание

Процедура TICKTOSTR преобразует дату Date, представленную во внутреннем формате, в ее символьный вид str в соответствии с заданными форматами преобразования DateFormat, YearFormat, TimeFormat и длиной символьной строки LengthStr.

Допустимые форматы преобразования:

1) DateFormat = 0 – дата с точкой ("dd.mm.yyyy");
2) DateFormat = 1 – дата с косой чертой ("dd/mm/yyyy");
3) YearFormat = 0 – вывод года без века ("dd.mm.yy");

 

4) `YearFormat = 1` – вывод года полностью("dd.mm.yyyy").

### Пример

```
typedef char Tstr[40];
...
INT   i;
DATE  Date;
Tstr  str;
...
/* Дата с точкой + год без века: */
TICKTOSTR (Date ,0 ,0 ,str, sizeof(Tstr));
...
```

## Форматное преобразование даты в строку

### Прототип

```
INT TICKTOSTRF (
   DECIMAL D1,    /* исходная дата */
   CHAR * F,      /* формат преобразования */
   CHAR * S);     /* выходная строка */
```

### Описание

Функция `TICKTOSTRF` преобразовывает дату `D1` из внутреннего представления в строку `S` согласно формату `F`. Для спецификации форматной строки можно использовать следующие обозначения:

- `+` – признак вывода интервала (+ или - или ' ');
- `DDD` – день года;
- `DD` – день месяца;
- `DY` – сокращенное название дня недели;
- `DAY` – название дня недели;
- `D` – номер дня недели;
- `MM` – номер дня в месяце;
- `MI` – минута часа;
- `MONTH` – название месяца;
- `MON` – сокращенное название месяца;
- `MS`, `FFF` – миллисекунда секунды;
- `YYYY` – год;
- `YY` – последние две цифры года;
- `HH24`, `HH` – час дня (от 0 до 23);
- `HH12` – час дня (от 1 до 12);
- `A.M. P.M. AM PM` – указатель времени до полудня или после полудня;

- SS – секунда минуты;
- FF – сотые доли секунды.

Если указан признак вывода знака интервала, то символьное представление интервала дат будет выдано с соответствующим знаком (+ или −), в противном случае интервал дат выводится без знака, и представление отрицательного интервала совпадает с представлением положительного.

**Примечание**
Интервалы лет и месяцев могут быть получены с помощью функции TICKTOSTRF с указанием шаблона, не содержащего других компонентов даты, кроме лет и месяцев.

### Примеры

Форматы даты:

```
"DD-Mon-YY", "DD-Mon-YYYY", "MM/DD/YY", "MM/DD/YYYY", "DD.MM.YY",
 "DD.MM.YYYY" " + MM/DD/YYYY".
```

Форматы времени:

```
"HH24", "HH24:MI", "HH24:MI:SS", "HH24:MI:SS.FF".
```

### Возвращаемое значение

1) 0 – успешное завершение.
2) 1 – ошибка преобразования.

### Пример

```
DECIMAL d;
CHAR s[8];
INT Error;
...
Error = TICKTOSTRF (d,"DD.MM.YY",s);
...
```

## Вычисление максимальной длины результата форматного вывода

### Прототип

```
INT TICKFSTRLEN (
   CHAR * F,       /* формат преобразования */
   INT FLEN);      /* длина строки формата */
```

### Описание

Функция TICKFSTRLEN вычисляет длину буфера, который необходимо резервировать под результат форматирования даты по формату, представленному в строке F с длиной FLEN. Строка формата F не обязательно должна заканчиваться нулевым символом. Спецификация формата приведена в подпункте «Форматное преобразование даты в

строку». Под полные названия дня и месяца должно быть зарезервировано по девять символов.

### Возвращаемое значение

Длина буфера, который необходимо зарезервировать под результат форматирования даты.

### Пример

```
INT Len;
...
Len=TICKFSTRLEN ("DD-MONTH-YYYY",strlen("DD-MONTH-YYYY"));
/* Должно быть возвращено значение 17 - 2 цифры день, */
/* максимум 9 цифр месяц, 4 цифры год и два символа "-" */
...
```

## Форматное преобразование строки в дату

### Прототип

```
INT STRTOTICKF (
   CHAR * S,      /* исходная строка */
   CHAR * F,      /* формат преобразования */
   DECIMAL D);    /* дата во внутреннем формате */
```

### Описание

Функция `STRTOTICKF` преобразует строку `S` в дату `D` согласно формату, представленному в строке `F`. Спецификация формата приведена в подпункте «Форматное преобразование даты в строку».

### Возвращаемое значение

1) `0` – успешное завершение.
2) `1`– ошибка преобразования.

### Пример

```
DECIMAL D;
INT Error;
...
Error = STRTOTICKF ("29|08|96","DD|MM|YY",D);
...
```

# Функции определения элементов даты

## Определение числа дней в месяце указанного года

### Прототип

```
INT DAYMON (
   INT Month, /* номер месяца */
   INT Year); /* номер года от начала н.э. */
```

### Описание

Функция DAYMON определяет количество дней (целое число от 28 до 31) в месяце Month (целое число в диапазоне от 1 до 12) указанного года Year (целое число в диапазоне от 1 до 9999).

### Возвращаемое значение

1) Количество дней в месяце – успешное завершение.
2) 0 – неправильные параметры.

### Пример

```
INT   i;
INT   Month;
INT   Year;
...
Year = 1994;
Month=    2;
i    = DAYMON (Month ,Year);
...
```

## Определение числа дней в году до указанной даты

### Прототип

```
INT DAYYEAR (
   INT Day,   /* номер дня в месяце */
   INT Month, /* номер месяца в году */
   INT Year); /* год от начала н.э. */
```

### Описание

Функция DAYYEAR вычисляет количество дней от начала текущего года до указанной даты, представленной номером дня Day (целое значение в диапазоне от 1 до 31), номером месяца (целое значение в диапазоне от 1 до 12) и номером года (целое значение в диапазоне от 1 до 9999).

### Возвращаемое значение

1) Вычисленное количество дней – успешное завершение.
2) 0 – неправильные параметры.

### Пример

```
INT   i;
INT   Day;
INT   Month;
INT   Year;
...
Year = 1994;
Month=    5;
Day  =   15;
```

 

```
i    = DAYYEAR (Day, Month, Year);
...
```

## Определение числа дней от начала н.э. до указанной даты

### Прототип

```
LONG DAYNUMBERDATE (
   INT Day,          /* номер дня в месяце */
   INT Month,        /* номер месяца в году */
   INT Year);        /* год от начала н.э. */
```

### Описание

Функция DAYNUMBERDATE вычисляет количество дней от начала н.э. до указанной даты, представленной номером дня Day (целое значение в диапазоне от 1 до 31), номером месяца (целое значение в диапазоне от 1 до 12) и номером года (целое значение в диапазоне от 1 до 9999).

### Возвращаемое значение

1) Вычисленное количество дней – успешное завершение.
2) 0 – неправильные параметры.

### Пример

```
LONG  i;
INT   Day;
INT   Month;
INT   Year;
...
Year  = 1994;
Month = 5;
Day   = 15;
i     = DAYNUMBERDATE (Day ,Month ,Year);
...
```

## Формирование даты по числу дней от начала н.э.

### Прототип

```
void DATEDAYNUMBER (
   LONG Ndate,       /* исходное число дней */
   INT * day,        /* день месяца */
   INT * month,      /* номер месяца */
   INT * year);      /* номер года */
```

### Описание

Функция DATEDAYNUMBER переводит заданное количество дней Ndate (положительное целое число типа LONG) в дату, отсчитываемую с начала н.э. и представляемую в виде трех чисел: день месяца day, номер месяца month и номер года year.

### Пример

```
INT   i;
LONG  Nday;
INT   day;
INT   month;
INT   year;
...
Nday = 790000;
i    = DATEDAYNUMBER (Nday ,&day ,&month ,&year);
...
```

## Определение дня недели по числу дней от начала н.э.

### Прототип

```
INT WEEKDAYNUMBER (
   LONG Date);       /* исходное число дней */
```

### Описание

Функция WEEKDAYNUMBER определяет номер дня недели (число от 1 до 7) по заданному количеству дней Date от начала н.э.

### Возвращаемое значение

1) Номер дня недели – успешное завершение.

2) 0 – неправильный параметр.

### Пример

```
INT   weekday;
LONG  Date;
...
Date   = 790000;
weekday= WEEKDAYNUMBER (Date);
...
```

## Проверка года на високосность

### Прототип

```
INT BIGYEAR (
   INT Year);  /* год от начала н.э. */
```

### Описание

Функция BIGYEAR проверяет, является ли указанный в Year год (целое значение от 1 до 9999) високосным или нет.

### Возвращаемое значение

1) 1 – год високосный.

 

2) 0 – год не високосный.

### Пример

```
INT   i;
INT   Year;
...
Year= 1994;
i   = BIGYEAR (Year);
...
```

# Аддитивные операции над датами

## Суммирование дат

### Прототип

```
void ADDDATE (
   DECIMAL D1,     /* первое слагаемое */
   DECIMAL D2,     /* второе слагаемое */
   DECIMAL dsumm); /* результат */
```

### Описание

Процедура ADDDATE складывает две даты D1 и D2, представленные в виде числа тиков от начала н.э., и помещает результат в переменную dsumm.

### Пример

```
INT   i;
DECIMAL  D, Ds;
...
/* Длину суток помещаем в 'Ds':        */
STRTOTICK ("02.01.0001:00:00:00.00" ,Ds);
/* Увеличиваем дату 'D' на одни сутки: */
ADDDATE (D ,Ds ,D);
...
```

## Вычитание месяцев из даты

### Прототип

```
void SUBMONTHSFROMDATE (
   DECIMAL D1,           /* исходная дата */
   DECIMAL D2,           /* кол-во вычитаемых месяцев */
   DECIMAL D3);          /* результат */
```

### Описание

Процедура SUBMONTHSFROMDATE вычитает из даты D1, представленной в виде числа тиков от начала н.э., число месяцев D2, представленных в виде числа тиков, и помещает

результат в переменную D3. Формат даты для D2 должен быть MM, MM.YYYY или MM.YY, где MM – число месяцев, YYYY – число вычитаемых лет.

## Добавление месяцев к дате

### Прототип

```
void ADDMONTHSTODATE (
   DECIMAL D1,          /* исходная дата */
   DECIMAL D2,          /* кол-во добавляемых месяцев */
   DECIMAL D3);         /* результат */
```

### Описание

Процедура ADDMONTHSTODATE складывает дату D1, представленную в виде числа тиков от начала н.э., и число месяцев D2, представленных в виде числа тиков, и помещает результат в переменную D3. Формат даты для D2 должен быть MM, MM.YYYY или MM.YY, где MM – число месяцев, YYYY – число добавляемых лет.

## Вычисление разницы между двумя датами

### Прототип

```
void SUBDATE (
   DECIMAL D1,  /* уменьшаемая дата */
   DECIMAL D2,  /* вычитаемая дата */
   DECIMAL D3); /* разность дат */
```

### Описание

Функция SUBDATE вычисляет разность дат D1 и D2, представленных в виде числа тиков от начала н.э., и помещает результат в переменную D3.

### Пример

```
DECIMAL D1, D2, D3;
...
/* Первая дата:          */
STRTOTICK ("01.01.0001:00:00:00.00" ,D1);
/* Вторая дата:          */
STRTOTICK ("02.01.0001:00:00:00.00" ,D2);
/* Длина суток в тиках: */
SUBDATE (D2 ,D1, D3);
...
```

## Увеличение интервала даты

### Прототип

```
void MULMONTHS  (
   DECIMAL D1,    /* интервал даты */
   DECIMAL D2,    /* множитель */
   DECIMAL D3);   /* произведение */
```

 

### Описание

Функция MULMONTHS умножает интервал лет и месяцев D1 на число D2, и помещает результат в переменную D3. Полученный результат усекается, при необходимости, до целого числа месяцев.

### Пример

```
DECIMAL  D1, D2, D3;
CHAR * Str = "0001.06"; /* 1 год и 6 месяцев */

/* Аргумент D1 – интервал лет и месяцев */
Error = STRTOTICKF (Str ,"YYYY.MM", D1);

if (Error == 0) ...
MULMONTHS  (D1,D2,D3);

/* Результат: D3 – интервал лет и месяцев, можно выбрать с помощью
 TICKTOSTRF */
```

## Уменьшение интервала даты

### Прототип

```
void DIVMONTH1 (
   DECIMAL D1,   /* интервал даты */
   DECIMAL D2,   /* делитель */
   DECIMAL D3);  /* частное */
```

### Описание

Функция DIVMONTH1 делит интервал лет и месяцев D1 на число D2, и помещает результат в переменную D3. Полученный результат усекается, при необходимости, до целого числа месяцев.

### Пример

```
DECIMAL  D1, D2, D3;
CHAR * Str = "0001.06"; /* 1 год и 6 месяцев */

/* Аргумент D1 – интервал лет и месяцев */
Error = STRTOTICKF (Str ,"YYYY.MM", D1);
if (Error == 0) ...
DIVMONTHS1  (D1,D2,D3);
/* Результат: D3 – интервал лет и месяцев, можно выдать с помощью
 TICKTOSTRF */
```

## Соотношение между интервалами дат

### Прототип

```
void DIVMONTH2 (
```

```
   DECIMAL D1,   /* интервал-делимое */
   DECIMAL D2,   /* интервал-делитель*/
   DECIMAL D3);  /* частное */
```

### Описание

Функция DIVMONTH2 вычисляет уровень превышения интервала-делимого D1 над интервалом-делителем D2 и помещает результат в переменную D3. Полученный результат возвращается без округления и усечения.

### Пример

```
DECIMAL  D1, D2, D3;
CHAR * Str1 = "0001.06"; /* 1 год и 6 месяцев */
CHAR * Str2 = "0002.10"; /* 2 года и 10 месяцев */


/* Аргумент D1 – интервал лет и месяцев */
Error = STRTOTICKF (Str1 ,"YYYY.MM", D1);

if (Error == 0) ...

Error = STRTOTICKF (Str2 ,"YYYY.MM", D2);
if (Error == 0) ...
DIVMONTHS2  (D1,D2,D3);
/* Результат: D3 – число DECIMAL, можно выдать с помощью DECTOSTR
 */
```

## Сравнение двух дат

### Прототип

```
INT CMPDATE (
   DECIMAL D1,  /* первая сравниваемая дата */
   DECIMAL D2); /* вторая сравниваемая дата */
```

### Описание

Функция CMPDATE сравнивает два даты D1 и D2, представленные во внутреннем формате даты.

### Возвращаемые значения

1) +1, если D1 > D2.
2) 0, если D1 = D2.
3) -1, если D1 < D2.

## Формирование даты

### Прототип

```
void DATETOTICK (
```

 

```
   LONG L1,       /* количество дней от начала н.э. */
   LONG L2,       /* время в тиках в последнем дне */
   DECIMAL D);    /* дата во внутреннем формате */
```

### Описание

Процедура DATETOTICK формирует дату D в стандартном виде по заданному числу дней L1 от начала н.э. и числу тиков L2 в последнем дне.

### Пример

```
LONG l1;
LONG l2;
DECIMAL D;
...
DATETOTICK (l1,l2,D);
...
```

# Функции редактирования

## Вставка в строку символа цифры

### Прототип

```
void PUTDIG (
   INT C,       /* вставляемая цифра */
   CHAR * S,    /* результирующая строка */
   INT * N,     /* позиция вставки */
   INT Ok);     /* флаг */
```

### Описание

Процедура PUTDIG помещает цифру из переменной C (если флаг **Ok** равен единице) или символ '?' (если флаг **Ok** равен нулю) в позицию N строки S и увеличивает указатель позиции N на единицу.

### Пример

```
CHAR str[3];
INT i = 0;
...
PUTDIG (i,str,&i,1);
PUTDIG (i,str,&i,0);
PUTDIG (i,str,&i,1);
...
```

## Вставка в строку двузначного числа в символьном виде

### Прототип

```
void PUT2 (
   INT X,     /* вставляемые цифры */
```

```
   CHAR * S, /* результирующая строка */
   INT * N,  /* позиция вставки */
   INT Ok);  /* флаг */
```

### Описание

Процедура PUT2 помещает две цифры числа из переменной X (если флаг **Ok** равен единице) или символы '??' (если флаг **Ok** равен нулю) в позицию N строки S и увеличивает указатель позиции N на два. Если число X не двузначное, результат не предсказуем.

### Пример

```
CHAR str[2];
INT i = 0;
INT x = 73;
...
PUT2 (x,str,&i,1);
...
```

## Вставка в строку трехзначного числа в символьном виде

### Прототип

```
void PUT3 (
   INT X,    /* вставляемые цифры */
   CHAR * S, /* результирующая строка */
   INT * N,  /* позиция вставки */
   INT Ok);  /* флаг */
```

### Описание

Процедура PUT3 помещает три цифры числа из переменной X (если флаг **Ok** равен единице) или символы '???' (если флаг **Ok** равен нулю) в позицию N строки S и увеличивает указатель позиции N на три. Если число X содержит более трех знаков, результат не предсказуем.

### Пример

```
CHAR str[3];
INT i = 0;
INT x = 169;
...
PUT3 (x,str,&i,1);
...
```

## Вставка в строку номера года в символьном виде

### Прототип

```
void PUTYEAR (
   INT Y,          /* вставляемые цифры года */
   INT ISCENTURY,  /* флаг представления года */
```

 

```
   CHAR * S,        /* результирующая строка */
   INT * N,         /* позиция вставки */
   INT Ok);         /* флаг */
```

### Описание

Процедура PUTYEAR помещает номер года из переменной Y в позицию N строки S. Если флаг **ISCENTURY** равен 0, то в строку будет помещён только год, и указатель позиции N увеличится на два. В противном случае в строку помещается век и год, и N увеличивается на четыре.

Если флаг **Ok** равен нулю, то в строку помещаются символы '??' или '????' соответственно значению флага **ISCENTURY**.

**Примечание**
Год необходимо указывать полностью, например, 1999.

### Пример

```
CHAR str[4];
INT i = 0;
INT x = 1996;
...
PUTYEAR (x,1,str,&i,1);
...
```

# Библиотека Int64

## Назначение

Библиотека Int64 предназначена для работы с длинными целыми числами (тип данных `BIGINT` СУБД ЛИНТЕР). Добавление этого типа данных вызвано необходимостью адресовать BLOB-данные, размеры которых могут достигать 64 Гбайтов.

Набор функций библиотеки обеспечивает выполнение:

1) арифметических операций;
2) операций сравнения чисел между собой;
3) логических операций.

## Условия применения

Библиотека может использоваться только в программах на языке C/C++.

Модуль языка C/C++, в котором предполагается использовать функции библиотеки, должен включать заголовочный файл `int64.h`.

В проект приложения должен быть добавлен файл `int64.c`.

**Примечание**
Файлы `int64.h` и `int64.c` входят в состав поставки СУБД ЛИНТЕР.

## Характеристики библиотеки

Длинные целые числа хранятся в полях размерностью 8 байт.

При выполнении арифметических операций ошибки переполнения не выявляются и соответствующие коды завершения не возвращаются.

На аппаратных платформах, обеспечивающих выполнение операций над длинными целыми числами, программные вызовы библиотечных функций будут автоматически заменяться их аппаратной реализацией.

## Функции

При описании функций библиотеки используются обозначения, определенные в заголовочном файле `int64.h`.

## Арифметические функции

### Сложение

#### Прототип

```
L_DLONG i64_Add (
   L_DLONG num1,   /* первое слагаемое */
```

```
   L_DLONG num2);  /* второе слагаемое */
```

### Описание

Функция i64_Add возвращает сумму двух длинных целых чисел, представленных в переменных num1 и num2 типа L_DLONG.

## Вычитание

### Прототип

```
L_DLONG i64_Sub (
   L_DLONG num1,   /* уменьшаемое */
   L_DLONG num2);  /* вычитаемое */
```

### Описание

Функция i64_Sub возвращает разность двух длинных целых чисел, представленных в переменных num1 и num2 типа L_DLONG.

## Умножение

### Прототип

```
L_DLONG i64_Mul (
   L_DLONG num1,   /* первый сомножитель */
   L_DLONG num2);  /* второй сомножитель */
```

### Описание

Функция i64_Mul возвращает произведение двух длинных целых чисел, представленных в переменных num1 и num2 типа L_DLONG.

## Целочисленное деление

### Прототип

```
L_DLONG i64_Div (
   L_DLONG num1,   /* делимое */
   L_DLONG num2);  /* делитель */
```

### Описание

Функция i64_Div выполняет целочисленное деление длинного целого числа, представленного в переменной num1, на другое длинное целое число, представленное в переменной num2. В случае если делитель равен нулю, функция возвращает максимальное значение типа L_DLONG.

## Получение остатка деления

### Прототип

```
L_DLONG i64_Rest (
   L_DLONG num1,   /* делимое */
   L_DLONG num2);  /* делитель */
```

### Описание

Функция i64_Rest выполняет целочисленное деление длинного целого числа, представленного в переменной num1, на другое длинное целое число, представленное в переменной num2, и возвращает остаток от этого деления. В случае если делитель равен нулю, функция возвращает случайное длинное целое число.

## Целочисленное деление с остатком

### Прототип

```
L_DLONG i64_DivRest (
   L_DLONG num1,       /* делимое */
   L_DLONG num2,       /* делитель */
   L_DLONG *rest);     /* адрес остатка */
```

### Описание

Функция i64_DivRest возвращает результат целочисленного деления длинного целого числа num1 на другое длинное целое число num2, а остаток от деления помещает по адресу, указываемому аргументом *rest, если он не NULL.

## Сложение целого и длинного целого

### Прототип

```
L_DLONG i64_AddLong (
   L_DLONG num1,      /* первое слагаемое */
   L_LONG num2);      /* второе слагаемое */
```

### Описание

Функция i64_AddLong возвращает сумму длинного целого числа, представленного в переменной num1 типа L_DLONG, и простого целого числа, представленного в переменной num2 типа L_LONG.

## Вычитание целого числа из длинного целого

### Прототип

```
L_DLONG i64_SubLong (
   L_DLONG num1,       /* уменьшаемое */
   L_LONG num2);       /* вычитаемое */
```

### Описание

Функция i64_SubLong возвращает разность длинного целого числа, представленного в переменной num1 типа L_DLONG, и простого целого числа, представленного в переменной num2 типа L_LONG.

## Умножение целого числа на длинное целое

### Прототип

```
L_DLONG i64_MulLong (
   L_DLONG num1,       /* первый сомножитель */
```

 

```
   L_LONG num2);       /* второй сомножитель */
```

### Описание

Функция i64_MulLong возвращает произведение длинного целого числа, представленного в переменной num1 типа L_DLONG, и простого целого числа, представленного в переменной num2 типа L_LONG.

## Целочисленное деление длинного целого числа на простое целое

### Прототип

```
L_DLONG i64_DivLong (
   L_DLONG num1,       /* делимое */
   L_LONG num2);       /* делитель */
```

### Описание

Функция i64_DivLong возвращает результат целочисленного деления длинного целого числа, представленного в переменной num1 типа L_DLONG, на простое целое число, представленное в переменной num2 типа L_LONG. В случае если делитель равен нулю, функция возвращает максимальное значение длинного целого числа.

## Получение остатка деления длинного целого числа на простое целое

### Прототип

```
L_DLONG i64_RestLong (
   L_DLONG num1,        /* делимое */
   L_LONG num2);        /* делитель */
```

### Описание

Функция i64_RestLong возвращает остаток целочисленного деления длинного целого числа, представленного в переменной num1 типа L_DLONG, на простое целое число, представленное в переменной num2 типа L_LONG. В случае если делитель равен нулю, функция возвращает случайное значение длинного целого числа.

# Логические функции

## Побитовое умножение

### Прототип

```
L_DLONG i64_And (
   L_DLONG num1,  /* первое число */
   L_DLONG num2); /* второе число */
```

### Описание

Функция выполняет побитовое умножение (логическая операция И) двух длинных целых чисел, представленных в переменных num1 и num2 типа L_DLONG.

## Побитовое сложение

### Прототип

```
L_DLONG i64_Or (
   L_DLONG num1,  /* первое число */
   L_DLONG num2); /* второе число */
```

### Описание

Функция выполняет побитовое сложение (логическая операция ИЛИ) двух длинных целых чисел, представленных в переменных num1 и num2 типа L_DLONG.

# Прочие функции

## Сравнение двух чисел

### Прототип

```
L_DLONG i64_Cmp (
   L_DLONG num1,  /* первое число */
   L_DLONG num2); /* второе число */
```

### Описание

Функция i64_Cmp сравнивает между собой значения двух длинных целых чисел, представленных в переменных num1 и num2 типа L_LONG.

### Возвращаемые значения

1) 1 – первый аргумент больше второго;
2) -1 – первый аргумент меньше второго;
3) 0 – аргументы равны.

## Изменение знака

### Прототип

```
L_DLONG i64_Neg (L_DLONG num);
```

### Описание

Функция i64_Neg изменяет знак длинного целого числа, представленного в переменной num типа L_DLONG, на противоположный.

## Проверка знака

### Прототип

```
 L_DLONG i64_IfNeg (L_DLONG num);
```

### Описание

Функция i64_ifNeg проверяет знак длинного целого числа, представленного в переменной num типа L_DLONG.

 

### Возвращаемые значения

1) 1 – число отрицательное;
2) 0 – число положительное.

## Проверка на равенство нулю

### Прототип

```
L_DLONG i64_IfZero (L_DLONG num);
```

### Описание

Функция i64_ifZero проверяет, является ли значение длинного целого числа, представленного в переменной num типа L_DLONG, равным нулю.

### Возвращаемые значения

1) 1 – если значение равно нулю;
2) 0 – в противном случае.

## Операция инкремента

### Прототип

```
L_DLONG i64_Inc (L_DLONG * num);
```

### Описание

Функция i64_Inc возвращает значение длинного целого числа, размещенного по адресу * num, увеличенное на единицу.

## Операция декремента

### Прототип

```
L_DLONG i64_Dec (L_DLONG * num);
```

### Описание

Функция i64_Dec возвращает значение длинного целого числа, размещенного по адресу * num, уменьшенное на единицу.

## Получение минимального знакового значения

### Прототип

```
L_DLONG i64_MINDLONG (void);
```

### Описание

Функция i64_MINDLONG возвращает минимальное знаковое значение длинного целого числа.

## Получение максимального знакового значения

### Прототип

```
L_DLONG i64_MAXDLONG (void);
```

### Описание

Функция `i64_MAXDLONG` возвращает максимальное значение длинного целого числа.

## Получение максимального беззнакового значения

### Прототип

```
L_DLONG i64_MAXUDLONG (void);
```

### Описание

Функция `i64_MAXUDLONG` возвращает максимальное беззнаковое значение длинного целого числа.

## Получение минимального беззнакового значения

### Прототип

```
L_DLONG i64_NILDL (void);
```

### Описание

Функция `i64_NILDL` возвращает минимальное беззнаковое значение длинного целого числа.

## Преобразование строки в число

### Прототип

```
L_DLONG i64_atol (L_CHAR *str);
```

### Описание

Функция `i64_atol` преобразует строку в значение длинного целого числа.

## Расширенное преобразование строки в число

### Прототип

```
L_DLONG i64_atol_ext (L_CHAR *str, L_WORD *err);
```

### Описание

Функция `i64_atol_ext` преобразует строку в значение длинного целого числа и возвращает код завершения в переменной `err`. Может вернуть три возможных кода:

1) `i64_NOERR` – нет ошибок;
2) `i64_ERR_NODIGIT` – строка не содержит число;
3) `i64_ERR_OVERFLOW` – произошло переполнение длинного целого.

 

## Преобразование числа в строку

### Прототип

```
L_CHAR * i64_ltoa (L_DLONG num, L_CHAR * str);
```

### Описание

Функция `i64_ltoa` преобразует длинное целое число `num` в строку `str` и возвращает указатель на строку `str`.

## Проверка на переполнение при преобразовании в unsigned long

### Прототип

```
L_BOOL i64_TestToLongOfl (L_DLONG num);
```

### Описание

Функция `i64_TestToLongOfl` возвращает `FALSE`, если число `num` при преобразовании в `unsigned long` вызовет переполнение.

## Проверка на переполнение при преобразовании в signed long

### Прототип

```
L_BOOL i64_TestToSLongOfl (L_DLONG num);
```

### Описание

Функция `i64_TestToSLongOfl` возвращает `FALSE`, если число `num` при преобразовании в `signed long` вызовет переполнение.

## Проверка на переполнение при преобразовании в double

### Прототип

```
L_BOOL i64_TestToDoubleOfl (L_DLONG num);
```

### Описание

Функция `i64_TestToDoubleOfl` возвращает `FALSE`, если число `num` при преобразовании в `double` вызовет переполнение.

## Проверка на переполнение при преобразовании из double

### Прототип

```
L_BOOL i64_TestFromDoubleOfl (double d);
```

### Описание

Функция `i64_TestFromDoubleOfl` возвращает `FALSE`, если число `d` при преобразовании в длинное целое вызовет переполнение.

## Инвертирование байтов

### Прототип

```
void i64_InvertBytes (L_DLONG * Val);
```

### Описание

Функция `i64_InvertBytes` меняет порядок байтов длинного целого числа, размещенного по адресу `* val`.

# Макросы

В состав библиотеки `int64` включен ряд макросов. В отличие от библиотечных функций, макросы в процессе трансляции исходного текста программы заменяются набором команд языка C/C++ и, возможно, некоторыми функциями библиотеки `int64`.

Список макросов приведен в таблице. Тексты макросов находятся в заголовочном файле `int64.h`.

Таблица. Список макросов

| Имя макроса | Назначение |
|---|---|
| `i64_ToLong(i,l)` | Преобразование простого целого числа в длинное целое |
| `i64_FromLong(l,i)` | Преобразование длинного целого числа в простое целое |
| `i64_ToDouble(i,d)` | Преобразование длинного целого числа в вещественное двойной точности |
| `i64_ToPosDouble(i,d)` | Преобразование длинного целого положительного числа в вещественное двойной точности |
| `i64_FromDouble(d,i)` | Преобразование вещественного числа двойной точности в длинное целое |
| `i64_FromPosDouble(d,i)` | Преобразование положительного вещественного числа двойной точности в длинное целое |
| `i64_GetLo(i)` | Выделение младших четырех байтов длинного целого числа |
| `i64_GetHi(i)` | Выделение старших четырех байтов длинного целого числа |
| `i64_Init(i,lo,hi)` | Присвоение значения длинному целому числу |
| `i64_IfPos(num)` | Проверка длинного целого числа на положительное значение |
| `i64_Equ(n1,n2)` | Проверка на равенство двух длинных целых чисел |

 

# Указатель функций библиотеки Decimals

# Указатель функций библиотеки Tick

# Указатель функций библиотеки Int64

## I