Краткий справочник по синтаксису языка программирования Free Pascal

Ввод и вывод данных

WRITELN(A,B,'SOME TEXT',C); – вывод значений переменных A,B, строки SOME TEXT и значения переменной C с переводом курсора на новую строку.
WRITE(A,B,'SOME TEXT',C); – то же что и WRITELN но без перевода курсора.
READLN(A,B,C); – ожидание ввода значений для переменных A,B,C с последующим переводом курсора на новую строку.
READ(A,B,C); – то же что и READLN, но без перевода курсора.
WRITE(A:5:1); – зарезервирует 5 знакомест под вывод значения A, и выведет его с точностью 1 знак после запятой. Справедливо для типов REAL и DOUBLE. Для WRITELN аналогично.
WRITE(A:3); – зарезервирует 3 знакоместа под вывод значения A, справедливо для целочисленных типов даных. Для WRITELN аналогично.

Типы данных

Объявление переменных идёт перед основной частью программы (перед первым BEGIN).
VAR A,B,C:INTEGER; E,F,SOMETHINGELSE:REAL; – объявление трёх переменных (A,B,C) целого типа, и ещё трех (E,F,SOMETHINGELSE) вещественного типа.
BOOLEAN – Логический тип, два значения TRUE или FALSE
BYTE – Целый тип, значения от 0 до 255
SMALLINT – Целый тип, значения от -128 до 127
INTEGER – Целый тип, значения от -32768 до 32767
LONGINT – Целый тип, значения по модулю не превышают ~2000000000
REAL – Вещественный тип, любые вещественные значения с достаточной точностью
DOUBLE – Вещественный тип, с повышеной точностью
CHAR – Символьный тип, значения от #0 до #255 (Справедливо для CP866 и прочих CODE PAGE)
STRING – Строковый тип, значения – любой текст, длиной до 255 символов. Также можно интерпретировать как массив символов.
TEXT – Переменная текстового файла.
ARRAY [A..B,C..D] OF TYPE – Двумерный массив типа TYPE, по первому измерению элементы нумеруются от A до B (обе должны быть целыми числами), по второму от C до D (соответственно). Пример: VAR A: ARRAY [1..10] OF INTEGER; – объявление массива из 10 переменных типа INTEGER;

Математические операторы

+ – Оператор сложения
- – Оператор вычитания
* – Оператор умножения
/ – Оператор деления (Результат только типа REAL или DOUBLE)
MOD – Оператор вычисления остатка от деления
DIV – Оператор целочисленного деления
ROUND() – Функция округления до ближайшего целого
RANDOM(N) Функция, возвращающая случайное целое от 0 до N-1; Для правильной работы необходимо перед вызовом RANDOM один раз вызвать RANDOMIZE();
SQR() – Функция вычисления квадрата числа
SQRT() – Функция вычисления корня второй степени числа
SIN(), COS(), TAN() – Тригонометрические функции
ARCTAN – Арктангенс
ARCTAN2 – Арктангенс от Y и X (Даёт знак угла, помимо модуля), содержится в модуле MATH

Модули

Подключение модулей производится перед объявлением переменных. В модулях содержатся дополнительные функции и процедуры, расширяющие базовую функциональность.
Пример USES PTCCRT,PTCGRAPH; – подключение модуля CRT и GRAPH.
PTCCRT – Модуль работы с псевдографикой, нажатиями клавиш, задержкой выполнения и т.п.
MATH – Модуль с математическими функциями
PTCGRAPH – Модуль для работы в графическом режиме
SYSUTILS – Модуль для работы с системными вызовами
Дополнительную информацию по модулям можно найти в каталоге !INFO в общем ресурсе SHARE.

Операции с логическими переменными и выражениями

A:=TRUE;
B:=5=4; //FALSE
TRUE OR TRUE // – TRUE
TRUE OR FALSE // – TRUE
FALSE OR FALSE // – FALSE
TRUE AND TRUE // – TRUE
TRUE AND FALSE // – FALSE
FALSE AND FALSE // – FALSE
TRUE XOR TRUE // – FALSE
TRUE XOR FALSE // – TRUE
FALSE XOR FALSE // – FALSE (порядок значения не имеет)
NOT TRUE //FALSE
NOT FALSE //TRUE

Условный оператор

Позволяет детализировать алгоритм и выполнить разные действия в зависимости от значения логического выражения.
IF (Логическое выражение) THEN (Действие #1) ELSE (Действие #2)
При значении TRUE логического выражения выполнится (Действие #1) в противном случае (Действие #2).
ELSE является необязательным параметром, т. е. Действие #1 может либо выполнятся, либо нет и без второго действия. Например: IF (Логическое выражение) THEN (Действие #1);
Перед ELSE точка с запятой не ставится!
При необходимости выполнить набор действий вместо одного их следует сгруппировать в операторные скобки BEGIN и END, тогда вся последовательность будет восприниматься как одно действие.

Циклы

Цикл FOR с инкрементом, переменная I (инкрементируемая переменная) присваивает значение равное A вначале цикла, и на каждой итерации цикла увеличивается на единицу. В каждой итерации цикла выполняется последовательность команд. Если команда всего одна, то BEGIN и END можно не ставить.
Цикл выполняется до тех пор, пока переменная I не станет равной B. Если B < A то цикл не выполнится ни одного раза.
FOR I:=A TO B DO
BEGIN
(Последовательность команд)
END;
Если необходимо декрементировать (уменьшать) переменную на единицу на каждой итерации то вместо TO следует писать DOWNTO
Замечания: I, A и B должны иметь целый тип. Переменную I менять внутри цикла нельзя.

Цикл WHILE с предусловием, выполняется до тех пор, пока условие (логическое выражение) принимает значение TRUE.
WHILE A>B DO
BEGIN
(Последовательность команд)
END;
Если изначально логическое выражение было ложным, цикл не выполнится ни разу. Во избежание «зацикливания», внутри цикла необходимо влиять на логическое выражение на котором основан цикл.

Цикл REPEAT … UNTIL с постусловием, выполняется до тех пор, пока условие (логическое выражение) не примет значение TRUE. Набор команд в BEGIN и END группировать необязательно. Если изначально логическое выражение было истинным, то цикл выполнится один раз (если внутри цикла его значение не изменится), т. к. условие проверяется после выполнения итерации.
REPEAT
(Последовательность команд)
UNTIL A=B;
Циклы можно вкладывать один в другой.
Процедура CONTINUE вызывает переход к следующей циклической итерации игнорируя расположенные ниже операторы, составляющие тело цикла. Процедура BREAK применяется для досрочного прекращения циклов.

Функции и процедуры

Функция – подпрограмма, выполняющая набор команд и в результате принимающая значение определённого типа. Аргумент функции – параметр, который передаётся внутрь функции для выполнения операций с ним. Функция может иметь множество аргументов.
Процедура – подпрограмма, выполняющая набор команд. Процедура также может иметь аргументы.

```
Процедура имя процедуры (аргументы:тип аргументов);
  объявление собственных переменных для процедуры
BEGIN
  исполняемая часть процедуры
END;

Функция имя функции (аргументы:тип аргументов):тип результата;
  объявление собственных переменных для функции
BEGIN
  исполняемая часть функции
END;
```

Для избежания путаницы в названиях глобальных переменных программы и внутренних переменных функции настоятельно рекомендуется указывать разные имена.
Процедура EXIT завершает работу своего программного блока. Если EXIT вызывается внутри процедуры или функции, то их работа завершается. Если EXIT вызывается в основном блоке программы, то это приводит к ее завершению.

Работа со строками

Строки можно присваивать друг другу. Строки можно объединять с помощью *операции конкатенации*, которая обозначается знаком +.

```
S1 := 'JOHN';
S2 := 'BLACK';
S1 := S1 + ' ' + S2;
```

К отдельному символу строки можно обращаться как к элементу массива символов, например S1[3]. Символ строки совместим с типом CHAR, их можно использовать в выражениях одновременно.
Функция COPY(S,START,LEN) возвращает подстроку длиной LEN, начинающуюся с позиции START строки S.
Процедура DELETE(S,START,LEN) удаляет из строки S, начиная с позиции START, подстроку длиной LEN.
Процедура INSERT(SUBS,S,START) вставляет в строку S подстроку SUBS, начиная с позиции START.
Функция LENGTH(S) возвращает фактическую длину строки S, результат имеет тип BYTE.
Функция POS(SUBS,S) ищет вхождение подстроки SUBS в строку S и возвращает номер первого символа SUBS в S или нуль, если SUBS не содержится в S.
Процедура STR(X,S) преобразует числовое значение X в строку S
Процедура VAL(S,X,ERRCODE) преобразует строку S в значение числовой переменной X, при этом строка SS должна содержать символьное представление числа. В случае успешного преобразования переменная ERRCODE равна нулю. Если же обнаружена ошибка, то ERRCODE будет содержать номер позиции первого ошибочного символа, а значение X не определено.

Комментарии и отступы в коде

Для того, чтобы Ваш код лучше читался его можно комментировать. Комментарии не обрабатываются компилятором и в них можно писать всё что угодно.
// – это однострочный комментарий, весь текст после двух слешей и до конца строки.

```
{Это
многострочный
комментарий, – Весь текст между фигурным скобками не обрабатывается компилятором}
```

Иногда для изменения части программы нужно убрать из неё часть кода. Вместо удаления этой части, можно поместить её в коментарий, добавив слеши и/или фигурные скобки. Тогда восстановить исходную версию будет проще.
Также с помощью комментариев можно объяснять что делает программа на этом участке, при их грамотном использовании читать и понимать работу программы гораздо проще.
Для повышения читаемости следует также пользоваться отступами от начала строки, которые будут показывать вложенность операторных скобок и команд в циклы или условные операторы.

Работа с текстовыми файлами

Для работы с файлом, необходимо выделить для него переменную типа TEXT и подключить эту переменную к файлу. Текстовый файл можно открыть либо только для чтения, либо только для записи.
ASSIGN(F,'FILENAME.TXT'); // – F – имя файловой переменной, FILENAME.TXT – имя файла (может содержать полный путь до файла, иначе будет браться путь где находится программа).
RESET(F); // – открытие файла, подключенного к файловой переменной F (далее файл F) для чтения данных из файла.
REWRITE(F); // – открытие файла F для записи данных в файл. Файл при этом перезаписывается, а всё что в нём содержалось до этого уничтожается.
WRITE(F,'SOME TEXT',A,B); // – запись в файл F, строки 'SOME TEXT' и переменных A и B также как и если бы это выводилось на экран. То же касается и WRITELN().
READ(F,A); //– Чтение значения переменной A из файла F, как если бы ввод производился с клавиатуры. То же касается и READLN().
EOF(F); //Функция, которая возвращает TRUE если достигнут конец файла F открытого для чтения, либо FALSE, если конец не достигнут.
EOLN(F), которая принимает значение TRUE, если конец текущей строки в файле F достигнут.
RENAME(F,'NEWNAME.TXT'); // – переименовывает файл F в NEWNAME.TXT
ERASE(F); // – Удаляет файл F
CLOSE(F); // – Закрывает файл F, необходимо выполнять перед концом работы с файлом.
С бинарными файлами, или файлами другого типа принцип работы схож.

PTCCRT

Тип	Имя	Описание
Процедура	**AssignCrt**	Связывает текстовый файл с окном CRT.
Процедура	**ClrEol**	Очищает все символы с позиции курсора до конца строки без перемещения курсора.
Процедура	**ClrScr**	Очищает экран и устанавливает курсор в верхний левый угол.
Процедура	**Delay**	Процедура задержки по таймеру.
Процедура	**DelLine**	Удаляет строку, содержащую курсор.
Процедура	**GotoXY**	Перемещает курсор в заданную позицию экрана.
Процедура	**HighVideo**	Устанавливает высокую интенсивность символов.
Процедура	**InsLine**	Вставляет пустую строку в позиции курсора.
Функция	**KeyPressed**	Определяет, была ли нажата клавиша на клавиатуре.
Процедура	**LowVideo**	Включает низкую интенсивность символов.
Процедура	**NormVideo**	Выбирает первоначальное значение атрибута текста.
Процедура	**NoSound**	Выключает внутренний динамик компьютера.
Функция	**ReadKey**	Читает символ из буфера клавиатуры.
Процедура	**Sound**	Включает внутренний динамик.
Процедура	**TextBackground**	Устанавливает цвет фона.
Процедура	**TextColor**	Выбирает цвет символов.
Процедура	**TextMode**	Устанавливает определенный текстовый режим.
Функция	**WhereX**	Возвращает X-координату текущего положения курсора.
Функция	**WhereY**	Возвращает Y-координату текущего положения курсора.
Процедура	**Window**	Определяет на экране текстовое окно.

SYSUTILS

Тип	Имя	Описание
Процедура	**Abort**	Прерывает обработку команд и выходит к последнему исключительному блоку
Функция	**AnsiCompareStr**	Сравнение двух строк на равенство
Функция	**AnsiCompareText**	Сравнение двух строк на равенство
Функция	**AnsiLowerCase**	Символы верхнего регистра изменяются в строку со строчными буквамм
Функция	**AnsiPos**	Находит позицию одной строки в другой
Процедура	**AppendStr**	Конкатенация одной строки в конец другой
Процедура	**Beep**	Делает звук гудка
Функция	**ChangeFileExt**	Изменяет расширение имени файла
Функция	**CompareStr**	Сравнивает две строки, чтобы увидеть, какая из них больше
Функция	**CompareText**	Сравнивает две строки, игнорируя регистр
Функция	**CreateDir**	Создаёт директорию
Переменная	**CurrencyDecimals**	Определяет число десятичных цифр в функции Format
Переменная	**CurrencyFormat**	Определяет размещение строки валюты в функции показа валюты
Переменная	**CurrencyString**	Строка валюты, используемая в функциях отображения валюты
Функция	**CurrToStr**	Преобразует денежную величину в строку
Функция	**CurrToStrF**	Преобразует денежную величину в строку с форматированием
Функция	**Date**	Возвращает текущую дату
Переменная	**DateSeparator**	Символ используемый для разделения полей отображаемой даты
Функция	**DateTimeToFileDate**	Преобразует значение TDateTime в формат date/time формат файла
Функция	**DateTimeToStr**	Конвертирует значение даты и времени TDateTime в строку
Процедура	**DateTimeToString**	Огромные возможности форматирования даты в строку
Функция	**DateToStr**	Преобразует значение даты TDateTime в строку
Функция	**DayOfWeek**	Выдает индекс дня недели для значения TDateTime
Переменная	**DecimalSeparator**	Символ используемый для отображения десятичной точки
Процедура	**DecodeDate**	Извлекает значения года, месяца, дня из TDateTime переменной
Процедура	**DecodeDateTime**	Разбивает TDateTime переменную на ее части даты/времени
Процедура	**DecodeTime**	Разбивает значение TDateTime на отдельные значения времени
Функция	**DeleteFile**	Удаляет файл, указанный в параметре
Функция	**DirectoryExists**	Возвращает true, если указанная директория существует
Функция	**DiskFree**	Выдает число свободных байтов на указанном диске
Функция	**DiskSize**	Выдает размер указанного диска в байтах
Функция	**EncodeDate**	Формирует значение TDateTime из значений года, месяца и дня
Функция	**EncodeTime**	Формирует значение TDateTime из значений часа, минуты, секунды и миллисеккунды
Функция	**ExtractFileDir**	Иизвлекает из полного имени файла название папки
Функция	**ExtractFileDrive**	Извлекает из полного имени файла название диска
Функция	**ExtractFileExt**	Извлекает из полного имени файла его расширение
Функция	**ExtractFileName**	Извлекает из полного имени файла краткое имя файла

Функция	**ExtractFilePath**	Извлекает из полного имени файла название патча
Функция	**FileAge**	Получение датя/время последнего изменения файла, не открывая его
Функция	**FileDateToDateTime**	Конвертирует формат даты/времени файла в значение TDateTime
Функция	**FileExists**	Возвращает True если указанный файл существует
Функция	**FileGetAttr**	Выдаёт атрибуты файла
Функция	**FileSearch**	Поиск файла в одной или более папках
Функция	**FileSetAttr**	Устанавливает атрибуты файла
Функция	**FindClose**	Закрывает успешный **FindFirst** поиск файла
Функция	**FindCmdLineSwitch**	Определяет, был передан некоторый параметр выключатель
Функция	**FindFirst**	Находит все файлы, соответствующие маске файла и атрибутов
Функция	**FindNext**	Находит следующий файл после успешного **FindFirst**
Функция	**FloatToStr**	Преобразует значение с плавающей запятой в строку
Функция	**FloatToStrF**	Преобразует значение с плавающей запятой в строку с форматированием
Функция	**ForceDirectories**	Создаёт новый путь каталогов
Функция	**Format**	Богатое форматирование чисел и текста в строке
Функция	**FormatCurr**	Богатое форматирование значений валюты в строку
Функция	**FormatDateTime**	Богатое форматирование переменной TDateTime в строку
Функция	**FormatFloat**	Богатое форматирование числа с плавающей запятой в строку
Процедура	**FreeAndNil**	Освобождение памяти объекта и установка его в nil
Процедура	**FreeMem**	Освобождает память, используемую переменной
Функция	**GetCurrentDir**	Возвращает текущий каталог (диск плюс каталог)
Процедура	**GetLocaleFormatSettings**	Получает региональные значения для безопасных потоков функций.
Функция	**IncMonth**	Увеличивает TDateTime переменную на некоторое число месяцев
Функция	**IntToHex**	Преобразует целое число в шестнадцатеричную строку
Функция	**IntToStr**	Конвертирует целое число в строку
Функция	**IsLeapYear**	Возвращает **True**, если данный календарный год високосный
Функция	**LastDelimiter**	Находит последнюю позицию указанных символов в строке
Переменная	**LongDateFormat**	Переводит длинную версию даты в строковый формат
Переменная	**LongDayNames**	Массив названий дней недели, начинается с 1 = Воскресенье
Переменная	**LongMonthNames**	Массив названий месяцев, начинается с 1 = Январь
Переменная	**LongTimeFormat**	Длинная версия времени в строковом формате
Функция	**LowerCase**	Изменяет символы верхнего регистра в строке в строчные буквы
Константа	**MinsPerDay**	Выдает число минут в дне
Константа	**MonthDays**	Выдает число дней в месяце
Переменная	**NegCurrFormat**	Определяет отображение отрицательного количества форматированной валюты
Функция	**Now**	Выдает текущую дату и время
Функция	**RemoveDir**	Позволяет удалить директорию
Функция	**Rename**	Переименовка фала
Функция	**RenameFile**	Переименование файла или директории
Процедура	**ReplaceDate**	Изменяет только часть даты TDateTime переменной

Процедура	**ReplaceTime**	Изменяет только часть времени TDateTime переменной
Константа	**SecsPerDay**	Выдает число секунд в дне
Функция	**SetCurrentDir**	Изменяет текущую директорию
Переменная	**ShortDayNames**	Массив названий дней недели, начиная с 1 = Воскресенье
Переменная	**ShortInt**	Целочисленный тип поддерживает значения от - 128 до 127
Переменная	**ShortMonthNames**	Массив названий дней месяца, начиная с 1 = Январь
Переменная	**ShortTimeFormat**	Короткая версия времени в строковый формат
Функция	**StrScan**	Ищет заданные символы в строке
Функция	**StrToCurr**	Преобразует числовую строку в денежное выражение
Функция	**StrToDate**	Конвертирует строку с датой в значение типа TDateTime
Функция	**StrToDateTime**	Конвертирует строку с датой и временем в значение типа TDateTime
Функция	**StrToFloat**	Преобразует числовую строку в значение с плавающей запятой
Функция	**StrToInt**	Преобразует строку с целым значением в Integer
Функция	**StrToInt64**	Преобразует строку с целым значением в Int64
Функция	**StrToInt64Def**	Преобразует строку с целым значением в Int64, учитывая значение по умолчанию
Функция	**StrToIntDef**	Преобразует строку с значение с типом Integer, учитывая значение по умолчанию
Тип	**TFloatFormat**	Форматы, используемые в функциях отображения чисел с плавающей запятой
Тип	**TFormatSettings**	Запись для содержания региональных значений для thread-safe функций
Переменная	**ThousandSeparator**	Символ, используемый для отображения разделителя тысяч
Функция	**Time**	Возвращает текущее время
Переменная	**TimeAMString**	Определяет значение AM в процедуре DateTimeToString
Переменная	**TimePMString**	Определяет значение PM в процедуре DateTimeToString
Переменная	**TimeSeparator**	Символ, используемый для разделения полей времени
Функция	**TimeToStr**	Конвертирует значение времени типа TDateTime в строку
Тип	**TReplaceFlags**	Определяет опции для подпрограммы **StringReplace**
Функция	**Trim**	Удаляет начальные и конечные пробелы в строке
Функция	**TrimLeft**	Удаляет начальные пробелы в строке
Функция	**TrimRight**	Удаляет конечные пробелы в строке
Тип	**TSearchRec**	Запись, используемая для хранения данных в **FindFirst** и **FindNext**
Тип	**TSysCharSet**	Символы, используемые снабженной строкой анализирующих функций
Переменная	**TwoDigitYearCenturyWindow**	Устанавливает порог столетия для преобразований строки года из 2 цифр
Функция	**UpperCase**	Изменяет символы в строке из нижнего регистра в верхний
Функция	**WrapText**	Добавьте перенос строки в строку, чтобы имитировать перенос слов

PTCGRAPH

Тип	Имя	Аргументы и результат	Описание
Процедура	**Arc**	(X, Y : Integer; StAngle, EndAngle, Radius : Word);	Рисует дугу окружности
Процедура	**Bar**	(X1, Y1, X2, Y2 : Integer);	Рисует закрашенный прямоугольник, используя текущие стиль и цвет закраски
Процедура	**Bar3D**	(X1, Y1, X2, Y2 : Integer; Depth : Word; Top : Boolean);	Рисует параллелепипед, используя текущий стиль и цвет закраски
Процедура	**Circle**	(X, Y : Integer; Radius : Word);	Рисует окружность текущим цветом, используя точку (X, Y) как центр
Процедура	**ClearDevice**	;	Очищает текущее устройство вывода и устанавливает текущий указатель в точку (0, 0)
Процедура	**ClearViewPort**	;	Очищает текущую область просмотра
Процедура	**CloseGraph**	;	Закрывает графическую систему
Процедура	**DetectGraph**	(Var GraphDriver, GraphMode : Integer);	Тестирует аппаратные средства и определяет, какой графический драйвер и режим можно использовать
Процедура	**DrawPoly**	(NumPoints : Word; Var PolyPoints);	Рисует контур многоугольника, используя текущий цвет и тип линии
Процедура	**Ellipse**	(X, Y : Integer; StAngle, EndAngle : Word; XRadius, YRadius : Word);	Рисует дугу эллипса
Процедура	**FillEllipse**	(X, Y : Integer; XRadius, YRadius : Word);	Рисует закрашенный эллипс (или круг, если радиусы одинаковы)
Процедура	**FillPoly**	(NumPoints : Word; Var PolyPoints);	Рисует закрашенный многоугольник
Процедура	**FloodFill**	(X, Y : Integer; Border : Word);	Закрашивает замкнутую область, используя текущие стиль и цвет закраски
Процедура	**GetArcCoords**	(Var ArcCoords:ArcCoordsType);	Возвращает координаты последней команды Arc
Процедура	**GetAspectRatio**	(Var XAsp, YAsp : Word);	Возвращает два числа, из которых может быть вычислен коэффициент сжатия
Функция	**GetBkColor**	: Word;	Возвращает текущий цвет фона
Функция	**GetColor**	: Word;	Возвращает текущий цвет
Процедура	**GetDefaultPalette**	(Var Palette : PaletteType);	Возвращает палитру, заданную по умолчанию
Функция	**GetDriverName**	: String;	Возвращает строку, содержащую имя текущего драйвера
Процедура	**GetFillPattern**	(Var FillPattern : FillPatternType);	Возвращает текущий шаблон закраски, установленный SetFillPattern
Процедура	**GetFillSettings**	(Var FillInfo : FillSettingsType);	Возвращает текущий цвет и шаблон закраски, установленные обращениями к процедурам SetFillPattern и SetFillStyle
Функция	**GetGraphMode**	: Integer;	Возвращает текущий графический режим
Процедура	**GetImage**	(X1, Y1, X2, Y2 : Integer; Var BitMap);	Сохраняет участок изображения в буфер
Процедура	**GetLineSettings**	(Var LineInfo : LineSettingsType);	Возвращает текущий тип, шаблон и толщину линии, установленные с помощью процедуры SetLineStyle
Функция	**GetMaxColor**	: Word;	Возвращает максимальный номер цвета, который может быть передан в процедуру SetColor
Функция	**GetMaxMode**	: Integer;	Возвращает максимальный номер доступного в настоящее время видеорежима

Функция	**GetMaxX**	: Integer;	Возвращает текущее значение разрешения по горизонтали
Функция	**GetMaxY**	: Integer;	Возвращает текущее значение разрешения по вертикали
Функция	**GetModeName**	(ModeNumber : Integer) : String;	Получает имя графического режима
Процедура	**GetModeRange**	(GraphDriver : Integer; Var LoMode, HiMode : Integer);	Возвращает диапазон допустимых значений графического режима для данного графического драйвера
Процедура	**GetPalette**	(Var Palette : PaletteType);	Возвращает текущую палитру и её размер
Функция	**GetPaletteSize**	: Integer;	Возвращает размер поисковой таблицы цветов палитры
Функция	**GetPixel**	(X, Y : Integer) : Word;	Возвращает значение пиксела в точке с координатами (X, Y)
Процедура	**GetTextSettings**	(Var TextInfo : TextSettingsType);	Возвращает установки для вывода текста в графическом режиме
Процедура	**GetViewSettings**	(Var ViewPort : ViewPortType);	Получает параметры текущей области просмотра
Функция	**GetX**	: Integer;	Возвращает X-координату текущего указателя (CP)
Функция	**GetY**	: Integer;	Возвращает Y-координату текущего указателя (CP)
Процедура	**GraphDefaults**	;	Обнуляет текущий указатель (CP), сбрасывает установки графической системы
Функция	**GraphErrorMsg**	(ErrorCode : Integer) : String;	Возвращает текст сообщения об ошибке по её номеру
Функция	**GraphResult**	: Integer;	Возвращает код ошибки для последней графической операции
Функция	**ImageSize**	(X1, Y1, X2, Y2 : Integer) : Word;	Возвращает число байт памяти, необходимых для сохранения заданной прямоугольной области экрана
Процедура	**InitGraph**	(Var GraphDriver : Integer; Var GraphMode : Integer; PathToDriver : String);	Инициализирует графическую систему и переводит видеокарту в графический режим
Функция	**InstallUserDriver**	(Name : String; AutoDetectPtr : Pointer) : Integer;	Добавляет внешний графический драйвер к таблице BGI драйверов
Функция	**InstallUserFont**	(FontFileName : String) : Integer;	Устанавливает новый шрифт
Процедура	**Line**	(X1, Y1, X2, Y2 : Integer);	Рисует линию из точки с координатами (X1, Y1) в точку с координатами (X2, Y2)
Процедура	**LineRel**	(Dx, Dy : Integer);	Рисует линию относительно текущего указателя (CP) и перемещает CP в точку с координатами (X1, Y1)
Процедура	**LineTo**	(X, Y : Integer);	Рисует линию от текущего указателя (CP) до (X, Y)
Процедура	**MoveRel**	(Dx, Dy : Integer);	Перемещает текущий указатель (CP) на заданное расстояние относительно его текущей позиции
Процедура	**MoveTo**	(X, Y : Integer);	Перемещает текущий указатель (CP) в точку с координатами (X, Y)
Процедура	**OutText**	(TextString : String);	Посылает строку на устройство вывода в позиции текущего указателя
Процедура	**OutTextXY**	(X, Y : Integer; TextString : String);	Посылает строку на устройство вывода
Процедура	**PieSlice**	(X, Y : Integer; StAngle, EndAngle, Radius : Word);	Рисует и закрашивает сектор окружности
Процедура	**PutImage**	(X, Y : Integer; var BitMap;	Помещает битовое изображение на экран

		BitBlt : Word);	
Процедура	**PutPixel**	(X, Y : Integer; Color : Word);	Ставит точку в позиции (X, Y)
Процедура	**Rectangle**	(X1, Y1, X2, Y2 : Integer);	Рисует прямоугольник, используя текущий тип и цвет линии
Процедура	**RegisterBGIdriver**	(Driver : Pointer) : Integer;	Передаёт графической системе указатель на драйвер устройства
Процедура	**RegisterBGIfont**	(Font : Pointer) : Integer;	Передаёт графической системе указатель на шрифт
Процедура	**RestoreCrtMode**	;	Восстанавливает первоначальный текстовый режим, такой, какой был перед инициализацией графического режима
Процедура	**Sector**	(X, Y : Integer; StAngle, EndAngle, XRadius, YRadius : Word);	Рисует и закрашивает сектор эллипса
Процедура	**SetActivePage**	(Page : Word);	Устанавливает активную страницу для графического вывода
Процедура	**SetAllPalette**	(Var Palette);	Изменяет все цвета в палитре на заданные
Процедура	**SetAspectRatio**	(Xasp, Yasp : Word);	Изменяет заданный по умолчанию коэффициент сжатия
Процедура	**SetBkColor**	(ColorNum : Word);	Устанавливает текущий цвет фона, используя палитру
Процедура	**SetColor**	(Color : Word);	Устанавливает текущий цвет, используя палитру
Процедура	**SetFillPattern**	(Pattern : FillPatternType; Color : Word);	Устанавливает определяемый пользователем шаблон закраски
Процедура	**SetFillStyle**	(Pattern : Word; Color : Word);	Устанавливает цвет и стиль закраски
Процедура	**SetGraphBufSize**	(BufSize : Word);	Изменяет заданный по умолчанию размер графического буфера, используемого для сканирования и заливки
Процедура	**SetGraphMode**	(Mode : Integer);	Устанавливает графический режим и очищает экран
Процедура	**SetLineStyle**	(LineStyle : Word; Pattern : Word; Thickness : Word);	Устанавливает текущий стиль линии
Процедура	**SetPalette**	(ColorNum : Word; Color : Shortint);	Заменяет один цвет в палитре
Процедура	**SetRGBPalette**	(ColorNum, RedValue, GreenValue, BlueValue : Integer);	Изменяет компоненты палитры для драйверов VGA и IBM 8514
Процедура	**SetTextJustify**	(Horiz, Vert : Word);	Устанавливает значения текстового выравнивания, используемые процедурами OutText и OutTextXY
Процедура	**SetTextStyle**	(Font, Direction : Word; CharSize : Word);	Устанавливает стиль вывода текста в графическом режиме
Процедура	**SetUserCharSize**	(MultX, DivX, MultY, DivY : Word);	Изменяет ширину и высоту символов для векторных шрифтов
Процедура	**SetViewPort**	(x1, y1, x2, y2 : Integer; Clip : Boolean);	Устанавливает текущую область просмотра или окно для графического режима
Процедура	**SetVisualPage**	(Page : Word);	Устанавливает номер видимой графической страницы
Процедура	**SetWriteMode**	(WriteMode : Integer);	Устанавливает режим записи для рисования линий
Функция	**TextHeight**	(TextString : String) : Word;	Возвращает высоту строки в пикселах
Функция	**TextWidth**	(TextString : String) : Word;	Возвращает ширину строки в пикселах.

MATH

Геометрические функции

Вызов	Операция
Hypot(x, y)	Гипотенуза треугольника с катетами x, y.
Norm(d)	Эвклидова норма массива d.

Определение максимума/минимума

Вызов	Описание
Max(a, b)	Максимальное целое из a, b.
MaxIntValue(d)	Максимальное целое из массива d.
MaxValue(d)	Максимальное значение из массива d.
Min(a, b)	Минимальное целое из a, b.
MinValue(d)	Минимальное значение из массива d.

Экспоненциальные и логарифмические функции

Вызов	Описание
Power(X, y)	X в степени y.
IntPower(X, y)	X в степени целого y.
LdExp(X, p)	Функция. X в степени 2 в степени p.
LnXp1(x: float):float	Функция. Натуральный логарифм от (x+1).
Log10(x: float):float	Функция. Логарифм x по основанию 10..
Log2(a,b:integer):integer	Функция. Логарифм x по основанию 2..
LogN(d:array of extended):extended	Функция. Логарифм x по основанию N..

Функции преобразования чисел

Вызов	Операция
Ceil(x:float):integer	Функция. Целое x округляется вверх.
Floor(x:float):integer	Функция. Целое x округляется вниз.
FrExp(x:float; var mant; var exp:integer)	Процедура. Возвращаются мантисса mant и порядок exp для вещественного x.

Функции преобразования единиц

Применяются для преобразования единиц измерения углов. Аргумент тригонометрических функций может исчисляться в:

- периодах,
- радианах (1 период = 2π радиан),
- градусах (1 период = 360 градусов),
- градах (1 период = 400 градов, используется в мореходстве).

Вызов	Операция
CycleToRad(x:float):float	Функция. Периоды в радианы.
DegToGrad(x:float):float	Функция. Градусы в грады.
DegToRad(x:float):float	Функция. Градусы в радианы.
GradToRad(x:float):float	Функция. Грады в радианы.
GradToDeg(x:float):float	Функция. Грады в градусы.
RadToCycle(x:float):float	Функция. Радианы в периоды.
RadToDeg(x:float):float	Функция. Радианы в градусы.
RadToGrad(x) :float):float	Функция. Радианы в грады.

Тригонометрические функции

Вызов	Операция
Tan(x: float):float	Функция. Тангенс от x.
Cotan(x: float):float Cot(x: float):float	Функция. Котангенс от x.
ArcSin(x: float):float	Функция. Обратный синус от x.
ArcCos(x: float):float	Функция. Обратный косинус от x.
ArcTan2(y,x: float):float	Функция. 4-х квадрантный обратный тангенс. ArcTan(y/x)
SinCos(x:float; out s,c: float):float	Процедура. s=Sin(x), c=Cos(x)

Гиперболические функции

Вызов	Операция
SinH(x:float):float	Функция. Синус гиперболический от x.
ArcSinH(x:float):float	Функция. Обратный синус гиперболический от x.
CosH(x:float):float	Функция. Косинус гиперболический от x.
ArcCosH(x:float):float	Функция. Обратный косинус гиперболический от x.
TanH(x:float):float	Функция. Тангенс гиперболический от x.
ArcTanH(x:float):float	Функция. Обратный тангенс гиперболический от x.

Статистические функции

Вызов	Операция
Mean(d:array of extended):float	Функция. Среднее значение массива d.
MeanAndStddev(d:array of extended; m,std:float)	Процедура. Среднее значение и стандартное отклонение массива d.
MomentsSkewKurtosis (d:array of extended; var m1,m2,m3,m4,skew,kurtosis:float)	Процедура. Для массива d возвращает 4 первых момента m1, m2, m3, m4, наклон skew, эксцесс kurtosis.
StdDev(d:array of extended):float	Функция. Стандартное отклонение массива d.
PopnStdDev(d:array of extended):float	Функция. Квадратный корень дисперсии массива d.
PopnVariance(d:array of extended; n:integer):float	Функция. Квадратный корень дисперсии из N значений массива d.
RandG(m,StdDev:float):float	Функция. Случайное число с Гауссовским распределением со средним значением m и стандартным отклонением StdDev
Sum(d:array of extended):float	Функция. Сумма значений массива d.
SumOfSquares(d:array of extended):float	Функция. Сумма квадратов значений массива d.
SumsAndSquares(d:array of extended;var s,ssqr):float	Процедура Суммы s и квадратов sqr значений массива d.
TotalVariance(d:array of extended):float	Функция. Дисперсия массива d.
Variance(d:array of extended):float	Функция. Дисперсия массива d.