Самарский государственный аэрокосмический университет
имени академика С.П. Королева

Статистические и финансовые функции Delphi

Самара 2004

Министерство образования Российской Федерации
Самарский государственный аэрокосмический университет
имени академика С.П. Королева

Статистические и финансовые функции Delphi

Методические указания

Самара 2004

Составитель: Стенгач М.С.

УДК 681.3.06

Статистические и финансовые функции Delphi: /Самар. гос. аэрокосм. ун-т; Сост. Стенгач М.С. Самара, 2004, 16 с.

В методических указаниях приводятся сведения о статистических и финансовых функциях популярной интерактивной среды программирования Delphi.

Предназначены для выполнения курсовых и лабораторных работ по курсу «Информатика» для студентов всех специальностей.

Составлены на кафедре "Компьютерные системы".

Печатаются по решению редакционно-издательского совета Самарского государственного аэрокосмического университета им. академика С.П. Королева.

Рецензент Г.С. Филин

В интерактивной среде программирования Delphi можно решать множество математических, статистических, финансовых и других задач, не создавая собственных программ для их реализации. Delphi содержит большое количество стандартных функций различного назначения. Стандартные функции размещаются в специальных модулях.

Все статистические и финансовые функции Delphi содержатся во вспомогательном модуле Math. Используя эти функции можно получить полную статистическую сводку по более чем тринадцати показателям.

В последнее время среде программирования Delphi посвящается большое количество учебной и методической литературы. При этом такому удобному и мощному средству, как стандартные статистические и финансовые функции, не уделяется почти никакого внимания. Однако все большее проникновение в финансовую и экономическую жизнь современных информационных технологий делает использование подобных средств все более актуальным.

1. Статистические функции

1.1. Функция Max

Функция `Max` возвращает наибольшее из двух чисел. Вызов функции выглядит следующим образом: `Max(A,B)`. Тип возвращаемого максимального значения `Max` (целый или вещественный) зависит от типа переменных `A` и `B`.

1.2. Функция Min

Функция `Min` возвращает наименьшее из двух чисел. Вызов функции выглядит следующим образом: `Min(A,B)`. Тип возвращаемого максимального значения `Min` (целый или вещественный) зависит от типа переменных `A` и `B`.

1.3. Функция MaxIntValue

Функция `MaxIntValue` возвращает наибольшее значение целочисленного массива `Data`.

Описание функции выглядит следующим образом:

```
function MaxIntValue(const Data: array of Integer): Integer;
```

1.4. Функция MaxValue

Функция `MaxValue` возвращает наибольшее значение вещественного массива `Data`.

Описание функции выглядит следующим образом:

```
function MaxValue(const Data: array of Double): Double;
```

1.5. Функция MinIntValue

Функция `MinIntValue` возвращает наименьшее значение целочисленного массива `Data`.

Описание функции выглядит следующим образом:

```
function MinIntValue(const Data: array of Integer): Integer;
```

1.6. Функция MinValue

Функция `MinValue` возвращает наименьшее значение вещественного массива `Data`.

Описание функции выглядит следующим образом:

```
function MinValue(const Data: array of Double): Double;
```

1.7. Функция Mean

Функция `Mean` возвращает среднее арифметическое всех значений массива чисел `Data`.

Описание функции выглядит следующим образом:

```
function Mean(const Data: array of Double): Extended;
```

1.8. Функция PopnVariance

Функция вычисляет дисперсию совокупности всех значений массива чисел `Data`. Дисперсия (от лат. dispersio – рассеяние) – в математической статистике и теории вероятности есть мера рассеивания (отклонения от среднего).

В статистике дисперсия есть среднее арифметическое из квадратов отклонений значений чисел a_1, a_2, …, a_n от их среднего арифметического a :

$$\sigma^2 = \frac{(a_1 - a)^2 + (a_2 - a)^2 + \cdots + (a_n - a)^2}{n}$$

где $a = \frac{a_1 + a_2 + \cdots + a_n}{n}$;

В теории вероятностей дисперсия случайной величины – математическое ожидание квадрата отклонения случайной величины от ее математического ожидания.

Описание функции:

```
PopnVariance (const Data: array of Double ): Extended;
```

1.9. Функция TotalVariance

Функция вычисляет статистическую дисперсию равную сумме квадратов разницы между каждым конкретным значением массива чисел a_1, a_2, …, a_n и их средним арифметическим значением a, т.е.

$$\sigma^2 = (a_1 - a)^2 + (a_2 - a)^2 + \cdots + (a_n - a)^2$$

Описание функции:

```
TotalVariance (const Data: array of Double ): Extended;
```

1.10. Функция Variance

Функция вычисляет статистическую типовую дисперсию всех значений числового массива `Data`, по методу n-1:

$$\sigma^2 = \frac{(a_1 - a)^2 + (a_2 - a)^2 + \cdots + (a_n - a)^2}{n-1}$$

Описание функции:

```
Variance(const Data: array of Double ): Extended;
```

1.11. Функция PopnStdDev

Функция вычисляет среднеквадратичное отклонение совокупности (квадратный корень дисперсии совокупности) всех значений массива `Data`.

Средним квадратичным отклонением называется квадратный корень из среднего арифметического всех квадратов разностей между данными числами и их средним арифметическим. Среднее квадратичное отклонение принято обозначать греческой буквой σ:

$$\sigma = \sqrt{\frac{(a_1 - a)^2 + (a_2 - a)^2 + \cdots + (a_n - a)^2}{n}}$$

Описание функции:

```
PopnStdDev (const Data: array of Double ): Extended;
```

1.12. Функция StdDev

Функция вычисляет стандартное среднеквадратичное отклонение (квадратный корень типовой дисперсии) значений массива чисел `Data`.

$$\sigma = \sqrt{\frac{(a_1 - a)^2 + (a_2 - a)^2 + \cdots + (a_n - a)^2}{n-1}}$$

Описание функции:

```
StdDev (const Data: array of Double ): Extended;
```

1.13. Процедура MeanAndStdDev

Процедура вычисляет среднее арифметическое `Mean` всех значений числового массива `Data` и среднее отклонение `StdDev` от среднего арифметического. Точность вычислений может быть потеряна, если среднее отклонение больше 10^7 или если числа массива отличаются на очень незначительную величину.

Описание процедуры:

```
MeanAndStdDev (const Data: array of Double;
               var Mean, StdDev: Extended );
```

1.14. Процедура MomentSkewKurtosis

Процедура вычисляет основные коэффициенты статистического анализа: первые четыре момента, коэффициент асимметрии и периодичность.

Описание процедуры:

```
MomentSkewKurtosis(const Data: array of Double;
                   var M1,M2,M3,M4,Skew,Kurtosis: Extended);
```

где

`M1` - среднее значение;

`M2` - дисперсия;

`Skew` - отклонение, отражающее симметрию распределения;

`Kurtosis` - периодичность, отражающая пологость распределения.

Пример:

```
Var Data: array of Double;
    M1, M2, M3, M4, Skew, Kur: Extended;
    I: Byte;
begin
  SetLength(Data, 10);
  for i:= 0 to 9 do Data[i]:= i;
  MomentSkewKurtosis(Data, M1, M2, M3, M4, Skew, Kur);
```

```
  { M1:=4.5; M2:=8.5; M3:=0; M4:=120.8625;
  Skew:=0;
  Kur:=1,77575757575758 }
end;
```

1.15. Функция Norm

Функция возвращает Евклидову норму для значений числового массива `Data`. Евклидова норма представляет собой квадратный корень суммы квадратов всех значений массива.

Описание функции:

```
Norm (const Data: array of Double ): Extended;
```

Пример:

```
Var Mas: array of Double;
    E: Extended;
    i: byte;
begin
  SetLength(Mas, 10);
  for i:= 0 to 9 do Mas[i]:= i;
  E:= Norm(Mas); // E:=16.8819430161341
end;
```

1.16. Функция Sum

Функция возвращает сумму значений всех элементов числового массива `Data`.

Описание функции:

```
Sum(const Data: array of Double ): Extended;
```

1.17. Функция SumInt

Функция возвращает сумму значений всех элементов целочисленного массива `Data`.

Описание функции:

```
SumInt(const Data: array of Integer ): Integer;
```

1.18. Функция SumOfSquares

Фнкция возвращает сумму квадратов всех значений числового массива `Data`:

Описание функции:

```
SumOfSquares (const Data: array of Double ): Extended;
```

1.19. Процедура SumsAndSquares

Процедура возвращает одновременно сумму и сумму квадратов всех значений числового массива `Data`.

Описание процедуры

```
SumsAndSquares (const Data: array of Double;
                var Sum, SumOfSquares: Extended );
```

1.20. Функция RandG

Функция генерирует случайное число с отклонением по Гауссу от среднего значения `Mean`. Наибольшая часть возвращаемых значений будет лежать в диапазоне `Mean-StdDev ... Mean+StdDev`.

Описание функции:

```
RandG ( Mean, StdDev: Extended ): Extended;
```

Пример:

```
Var X: Real;
begin
  Randomize;
  X:= RandG( 100, 10 );
  {Наибольшее количество полученных значений X будет
   находиться в интервале 90<=X<=110 }
end;
```

2. Финансовые функции

2.1. Функция DoubleDecliningBalance

Функция вычисляет амортизационные отчисления на определенном этапе. В качестве параметров используются: начальная стоимость `Cost`, срок службы `Life`, конечная стоимость `Salvage` и амортизационный период `Period`.

Описание функции:

```
DoubleDecliningBalance (Cost, Salvage: Extended;
                        Life, Period: Integer ): Extended;
```

Вычисления производятся по следующим формулам:

Период	Сумма амортизационных отчислений
1	A1:= Cost*2/Life
2	A2:=(Cost-A1)*2/Life
3	A3:=(Cost-A1-A2)*2/Life
...	...
N	An:=(Cost-A1-A2-…-An-1)*2/Life, где n=Period

При достижении конечной стоимости, амортизация перестает начисляться, и для последующих периодов функция будет возвращать значение, указанное в параметре `Salvage`.

Пример

```
Var A : Extended;
begin
  A:=DoubleDecliningBalance(1000,100,5,1); // A:=400
end;
```

2.2. Функция FutureValue

Функция вычисляет значение вклада по прошествии определенного периода времени.

Описание функции:

```
FutureValue (Rate: Extended;
             NPeriods: Integer ;
             Payment, PresentValue: Extended;
             PaymentTime: TPaymentTime ): Extended;
```

где

`Rate` - дивиденды, начисляемые за единицу периода (процентная ставка вклада);

`NPeriods` - количество прошедших единиц периода времени. Например, если проценты начисляются раз в год, то данный параметр определяет количество лет;

`PresentValue` - первоначальная сумма вклада.

В параметре `Payment` указывается сумма, которая будет добавляться вкладчиком к вкладу в течение каждой единицы периода времени. Если вклад осуществляется только один раз, то `Payment:=0`.

Параметр `PaymentTime` определяет, как должны начисляться проценты:

Значение	Описание
`PtStartOfPeriod`	Проценты начисляются в конце единицы периода времени на всю сумму, находящуюся на момент начисления на счету.
`PtEndOfPeriod`	Проценты начисляются только на сумму, которая находилась на счету в начале единицы периода времени. Т.е. на сумму, которая была добавлена вкладчиком к вкладу в течение последней единицы времени, проценты будут начислены по прошествии следующего периода.

Знак минус в результате функции показывает, что данная сумма является кредитовой, т.е. должна быть списана со счета.

Начисление процентов происходит по схеме сложных процентов, т.е. по мере начисления процентов учитываются предыдущие процентные начисления. Например, если исходная сумма вклада равна P, величина периодических платежей – A, а процентная ставка – R, то наращиваемый денежный поток F будет равен:

к концу первого периода времени:

$F_1=P+A+P\cdot R= P\cdot(1+R)+A$ – если проценты начисляются в конце периода;

$F_1=P+A+(P+A)\cdot R=(P+A)\cdot(1+R)$ – если проценты начисляются в начале периода.

к концу второго периода времени:

$F_2=F_1+A+F_1 \cdot R=F_1 \cdot (1+R)+A= P \cdot (1+R)^2+A \cdot (1+R)+A$ – если проценты начисляются в конце периода;

$F_2=F_1+A+(F_1+A) \cdot R=(P+A) \cdot (1+R)^2+A \cdot (1+R)$ – если проценты начисляются в начале периода.

к концу третьего периода времени:

$F_3=F_2+A+F_2 \cdot R=P \cdot (1+R)^3+A \cdot ((1+R)^2+(1+R)+1)$ – если проценты начисляются в конце периода;

$F_3=F_2+A+(F_2+A) \cdot R=(P+A) \cdot (1+R)^3+A \cdot ((1+R)^2+(1+R))$ – если проценты начисляются в начале периода.

Конечная формула для определения величины вклада по прошествии n периодов времени принимает вид:

$$F_n = P \cdot (1+R)^n + A \cdot \sum_{k=0}^{n-1} (1+R)^k$$ – если проценты начисляются в конце периода;

$$F_n = (P+A) \cdot (1+R)^n + A \cdot \sum_{k=1}^{n-1} (1+R)^k$$ – если проценты начисляются в начале периода.

Пример:

Первоначальный вклад составил 100 руб., дивиденды - 10% годовых. Ежегодно вкладчик добавляет к вкладу 50 руб. Проценты начисляются в конце года только на сумму, которая находилась на счету в начале года, т.е. проценты на сумму, которая была добавлена к вкладу в течение года будут начислены только в конце следующего года. Вычисляем сумму вклада через 5 лет.

```
Var Sum : Extended;
begin
  Sum:= FutureValue(0.1, 5, 50, 100, ptEndOfPeriod);
  // Sum:= -466.306
end;
```

2.3. Функция InterestPayment

Функция вычисляет процентную ставку кредита на определенном этапе в денежном исчислении.

Описание функции:

```
InterestPayment (Rate: Extended;
                 NPeriods: Integer ;
                 PresentValue, FutureValue: Extended;
                 PaymentTime: TPaymentTime ): Extended;
```

где

`PresentValue` - сумма кредита;

`Rate` - фиксированная процентная ставка кредита;

`NPeriods` - число этапов, в течение которых производятся выплаты;

`Period` - номер этапа выплат, для которого производятся вычисления;

`FutureValue` - значение суммы кредита, по истечении выплат.

Параметр `PaymentTime` определяет, происходят выплаты в начале (`PtStartOfPeriod`) этапа или в конце (`ptEndOfPeriod`).

Знак минус в результате функции показывает, что данная сумма является кредитовой, т.е. должна быть списана со счета.

Пример

```
Var Pay : Extended;
begin
  Pay:=InterestPayment( 0.1, 1, 5, 1000, 0, PtStartOfPeriod);
  // Pay:= -90.9090909090909
end;
```

2.4. Функция InterestRate

Функция вычисляет процентную ставку инвестиций, необходимую для возврата суммы инвестиций `PresentValue` с дивидендами.

Описание функции:

```
InterestRate (NPeriods: Integer;
              Payment, PresentValue, FutureValue: Extended;
              PaymentTime: TPaymentTime ): Extended;
```

где

`PresentValue` - сумма инвестиций;

`FutureValue` - полная сумма, полученная от инвестиций. Включает возврат первоначальной суммы инвестиций и дивиденды;

`NPeriods` - количество этапов выплат;

`Payment` - сумма периодических выплат;

Параметр PaymentTime определяет, происходят выплаты в начале (PtStartOfPeriod) этапа или в его конце (ptEndOfPeriod).

Пример

```
Var IRate : Extended;
begin
  IRate:=InterestRate(1, -100, -1000, 1500, ptEndOfPeriod);
  { IRate:= 0.4 }
end;
```

Следует аккуратно использовать финансовые функции и внимательно следить за расстановкой знаков плюс и минус в параметрах функций.

2.5. Функция InternalRateOfReturn

Функция вычисляет внутреннюю процентную ставку дохода от инвестиций. Массив CashFlows состоит из значений инвестиций и значений получаемого дохода за определенные периоды времени. Первое значение массива должно быть отрицательным, так как оно определяет первоначальную сумму инвестиций. Последующие значения могут быть отрицательными (дополнительные инвестиции), положительными (получаемый доход) или равны 0.

Описание функции:

```
InternalRateOfReturn (
Guess: Extended;
const CashFlows: array of Double ): Extended;
```

Пример

```
Var IRate: Extended;
CashFlows: array of Double;
begin
  SetLength(CashFlow, 2);
  CashFlow[0]:=-1000;
  CashFlow[1]:= 1200;
  IRate:= InternalRateOfReturn(0, CashFlow);
  // IRate:= 0.4
end;
```

2.6. Функция NetPresentValue

Функция вычисляет значение текущего платежа, используя массив с расчетными значениями. Данная функция помогает определить расчетную стоимость инвестиций на основе предполагаемого (расчетного) дохода. Параметр Rate определяет процентную ставку инвестиций, CashFlows - массив расчетных значений текущих платежей. Параметр PaymentTime указывает, происходят выплаты в начале (PtStartOfPeriod) или в конце (ptEndOfPeriod) платежного этапа.

Описание функции:

```
NetPresentValue (Rate: Extended;
                 const CashFlows: array of Double;
                 PaymentTime: TPaymentTime ): Extended;
```

Пример

```
Var PayValue: Extended;
CashFlows: array of Double;
begin
SetLength(CashFlow, 3);
CashFlow[0]:=-100;
CashFlow[1]:= 110;
CashFlow[2]:= 121;
PayValue:= NetPresentValue (0.1, CashFlow, PtStartOfPeriod);
{ PayValue:= 100 }
end;
```

2.7. Функция NumberOfPeriods

Функция определяет количество этапов, необходимых для погашения кредита до значения определенного в параметре FutureValue.

Описание функции:

```
NumberOfPeriods(
Rate, Payment, PresentValue, FutureValue: Extended;
PaymentTime: TPaymentTime ): Extended;
```

где

PresentValue - первоначальная сумма кредита;

Rate - процентная ставка;

`Payment` - величина регулярных выплат.

Параметр `PaymentTime` определяет, происходят выплаты в начале (`PtStartOfPeriod`) этапа или в его конце (`ptEndOfPeriod`).

Пример

Первоначальная сумма кредита составляет 364руб. Процентная ставка 20%. Регулярные ежемесячные выплаты по 100руб производятся в конце месяца. Рассчитаем сколько месяцев необходимо для полной выплаты кредита.

```
Var Np: Extended;
begin
  Np:= NumberOfPeriods ( 0.2, 100, 364, 0, ptEndOfPeriod);
  { Np:= -3 }
end;
```

2.8. Функция Payment

Функция вычисляет общую сумму погашения кредита, выплачиваемую на каждом этапе (состоит из текущих выплат и процентов).

Описание функции:

```
Payment (Rate: Extended;
         NPeriods: Integer;
         PresentValue, FutureValue: Extended;
         PaymentTime: TPaymentTime ): Extended;
```

где

`PresentValue` - сумма заимствования.

`NPeriods` - срок выплаты кредита. Данный параметр указывается количество этапов выплаты (количество лет, кварталов, месяцев и т.д.).

`FutureValue` - оставшаяся сумма кредита по истечении указанного периода.

`Rate` - величина процентной ставки кредита (ежегодная, ежеквартальная, ежемесячная и т.д. в соответствии с единицей измерения периода времени).

Параметр PaymentTime определяет, как происходят платежи: в начале (PtStartOfPeriod) или в конце (ptEndOfPeriod) платежного периода.

Пример

```
Var PaySum : Extended;
begin
  PaySum:= Payment(0.2, 1, 100, 0, ptEndOfPeriod);
  { PaySum:=-120 }
end;
```

2.9. Функция PeriodPayment

Функция вычисляет сумму выплат основной части кредита (без учета процентов) на определенном этапе.

Описание функции:

```
PeriodPayment (Rate: Extended;
               Period, NPeriods: Integer;
               PresentValue, FutureValue: Extended;
               PaymentTime: TPaymentTime ): Extended;
```

где

`PresentValue` - сумма заимствований;

`NPeriods` - срок выплаты кредита (количество этапов);

`FutureValue` - сумма кредита по прошествии указанного срока;

`Period` - номер этапа, для которого производятся вычисления;

`Rate` - процентная ставка кредита.

Параметр `PaymentTime` определяет, как происходят платежи: в начале (`PtStartOfPeriod`) или в конце (`ptEndOfPeriod`) платежного периода.

Сумму выплачиваемых процентов кредита можно вычислить с помощью функции **InterestPayment.**

Пример

```
Var PaySum : Extended;
begin
  PaySum:= PeriodPayment( 0.1, 2, 5, 1000, 0, ptEndOfPeriod);
  { PaySum:= -180.17722887422 }
end;
```

2.10. Функция PresentValue

Функция определяет значение вклада в указанный период времени.

Описание функции:

```
PresentValue (Rate: Extended;
              NPeriods: Integer;
              Payment, FutureValue: Extended;
              PaymentTime: TPaymentTime ): Extended;
```

где

`Payment` - первоначальная сумма вклада;

`NPeriods` - срок вклада;

`Rate` - процентная ставка;

`FutureValue` - значение, которого могут достигнуть инвестиции в определенный период.

Параметр `PaymentTime` указывает, как происходят платежи: в начале (`PtStartOfPeriod`) или в конце (`ptEndOfPeriod`) платежного периода.

Пример

```
Var Value : Extended;
begin
  Value:= PresentValue( 0.1, 1, -100, 0, ptEndOfPeriod);
  { Value:= 90,9090909090909 }
end;
```

2.11. Функция SLNDepreciation

Функция вычисляет сумму амортизационных отчислений за единицу периода времени по методу линейной (равномерной) амортизации.

Описание функции:

```
SLNDepreciation (Cost, Salvage: Extended;
                 Life: Integer ): Extended;
```

где

`Cost` - первоначальную стоимость оборудования;

`Salvage` - конечная стоимость оборудования;

`Life` - срок эксплуатации.

2.12. Функция SYDDepreciation

Функция вычисляет сумму амортизационных отчислений на заданном этапе по методу ускоренной амортизации.

Описание функции:

```
SYDDepreciation (Cost, Salvage: Extended;
                 Life, Period: Integer ): Extended;
```

где `Period` - номер этапа, для которого определяется сумма амортизационных отчислений.

Пример

```
Var Amort: Extended;
begin
  Amort:= SYDDepreciation( 1000, 100, 5, 1); { Amort:=300 }
end;
```

Содержание

Учебное издание

СТАТИСТИЧЕСКИЕ И ФИНАНСОВЫЕ ФУНКЦИИ DELPHI

Составитель: *Стенгач Михаил Сергеевич*

Редактор Н.С. Куприянова

Подписано в печать . .2004 г. Формат 60x84 1/16

Бумага офсетная. Печать офсетная.

Усл. печ.л. Усл.кр.-отт. Уч.-изд.л.

Тираж 100 экз. Арт. С-3(Д5)/2003. Заказ

Самарский государственный аэрокосмический университет имени академика С.П. Королева.

443086 Самара, Московское шоссе, 34.

ИПО СГАУ 443001 Самара, ул. Молодогвардейская, 151.