|  |  |
| --- | --- |
| Ограничение | Действие ограничения целостности |
| **data type**  тип данных | Предотвращает появление в столбце значений, не соответствующих типу данных |
| **not null**  запрет значений **null** | Предотвращает появление в столбце значений null |
| **default**  знач. по умолчанию | Устанавливает значение в столбце по умолчанию при выполнении операции INSERT |
| **primary key**  первичный ключ | Предотвращает появление в столбце повторяющихся значений и пустого значения |
| **foreign key**  внешний ключ | Устанавливает связь между таблицей со столбцом, имеющим свойство **foreign key** и таблицей, имеющей столбец со свойством **primary key**; |
| **unique**  уникальное значение | Не допускает пустые и повторяющиеся значения, не может быть использовано для связи с полем другой таблицы |
| **check**  проверка значений | Предотвращает появление в столбце значения, не удовлетворяющего логическому условию |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| MSS поддерживает следующие типы данных: числовые, символьные, для хранения даты и времени, денежные, двоичные и специальные.   | **Тип данных** | **Описание** | **Тип данных** | **Описание** | | --- | --- | --- | --- | | **integer**  **(int)** | Целочисленные значения, занимают 4 байта. | **date** | Дата, занимает 3 байта | | **smallint** | Целочисленные значения, занимают 2 байта | **time[(p)]**  **0 ≤ p ≤ 7** | Время, занимает от 3 до 5 байтов. **p** − колич. знаков после точки в секундах | | **tinyint** | Неотрицательные целочисленные значения, занимают 1 байт | **smalldatetime** | Дата и время, занимает 2 байта | | **bigint** | Целочисленные значения, занимают 8 байт | **datetime** | Дата и время, занимает 4 байта | | **real** | Вещественные числа с плавающей точкой | **datetime2** | Дата и время, занимает 8 байтов | | **decimal (p[,s] )**  **(dec)**  или  **numeric (p[,s] )** | Вещественные значения с фикс. точкой, **р** − общее количество цифр, **s** − количество цифр после точки. Занимает от 5 до 17 байт. | **сhаr[(n)]** | Строки фиксированной длины из однобайтовых симво­лов, n − количество символов | | **float[(p)]** | Вещественные числа с плавающей точкой. Если **р** < 25 − одинарная точность (4 байта), при **р** > 25 − двойная точность (8 байтов) | **nchar[(n)]** | Строки фиксированной длины символов Unicode. Каждый символ занимает 2 байта. | | **money** | Денежные значения, занимают 8 байтов | **varchar[(n)]** | Строки переменной длины из однобайтовых симво­лов | | **smallmoney** | Денежные значения, занимают 4 байта | **nvarchar[(n)]** | Строки переменной длины символов Unicode − 2 байта | | **binary(n)** | Задает битовую строку, длиной ровно **n** байтов | **varbinary(n)** | Задает битовую строку, длиной не более **n** байтов | |

Переменные в MS SQL Server представляют собой именованные хранилища, которые используются для временного хранения данных в рамках текущей сессии работы с базой данных. Они могут содержать различные типы данных, такие как числа, строки, даты и другие.

Для объявления переменной в MS SQL Server используется ключевое слово DECLARE, например:  
DECLARE @myVariable INT;

В этом примере мы объявляем переменную @myVariable типа INT (целое число).

Для присвоения значения переменной используется оператор присваивания "=", например:

SET @myVariable = 42;

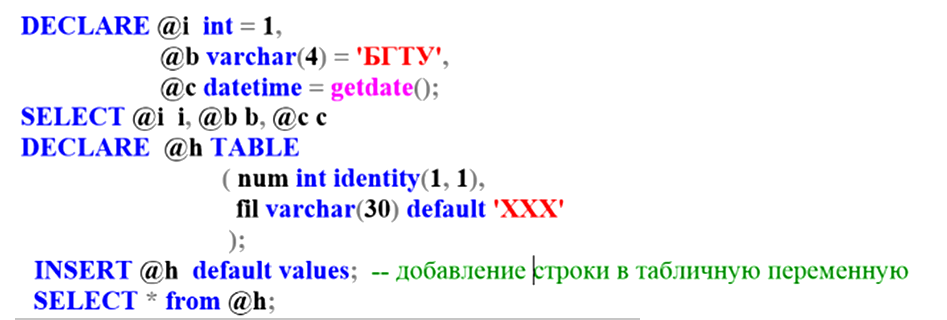
В этом примере мы присваиваем переменной @myVariable значение 42.

Переменные в MS SQL Server могут использоваться для хранения промежуточных результатов вычислений, передачи значений между различными запросами, управления ходом выполнения хранимых процедур и функций, а также для более удобного форматирования запросов.

Для объявления переменных, используемых в программах, предназначен оператор DECLARE

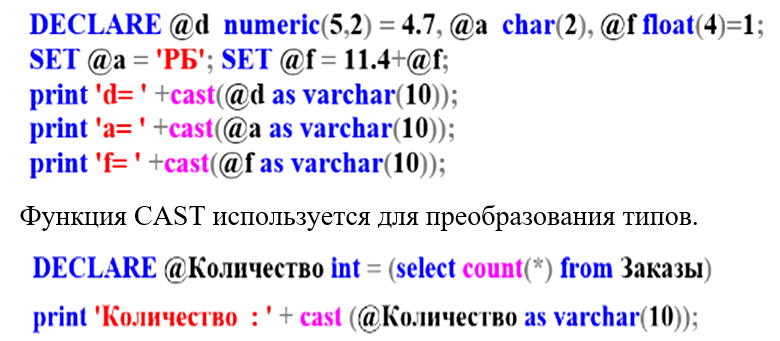
Имя переменной должно начинаться с символа @.

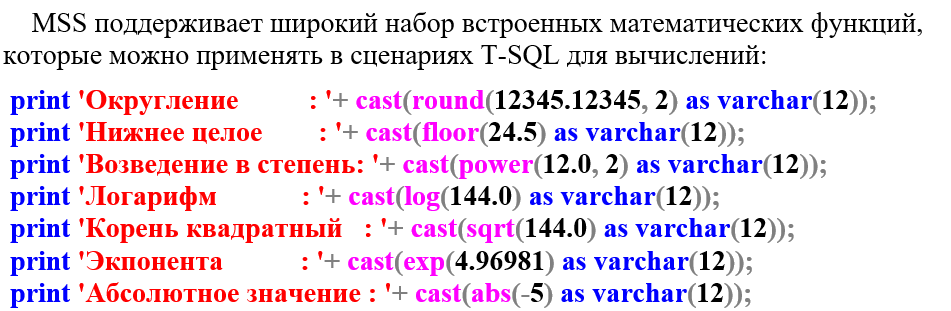
С помощью оператора SET можно одной переменной присвоить значение и выполнять вычисления. Оператор SELECT позволяет нескольким переменным присвоить значения.

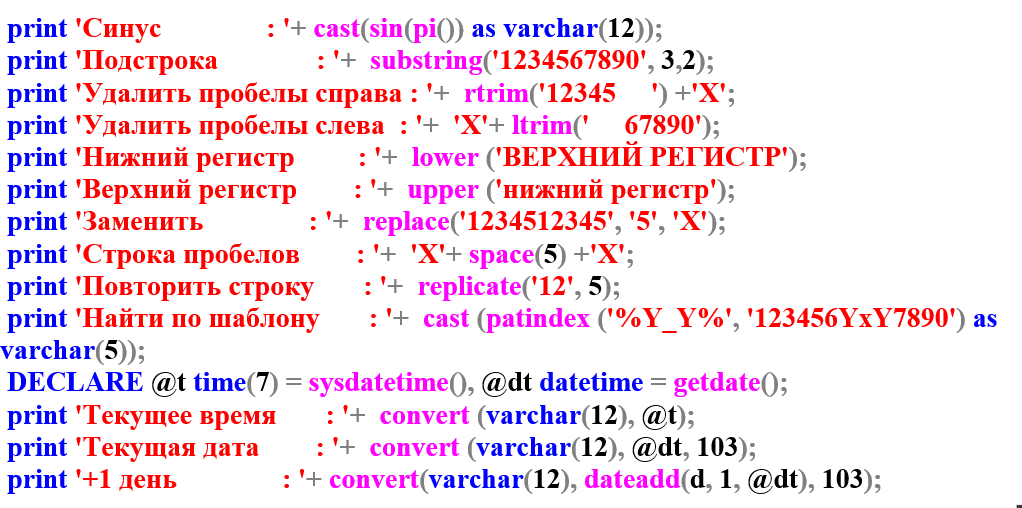


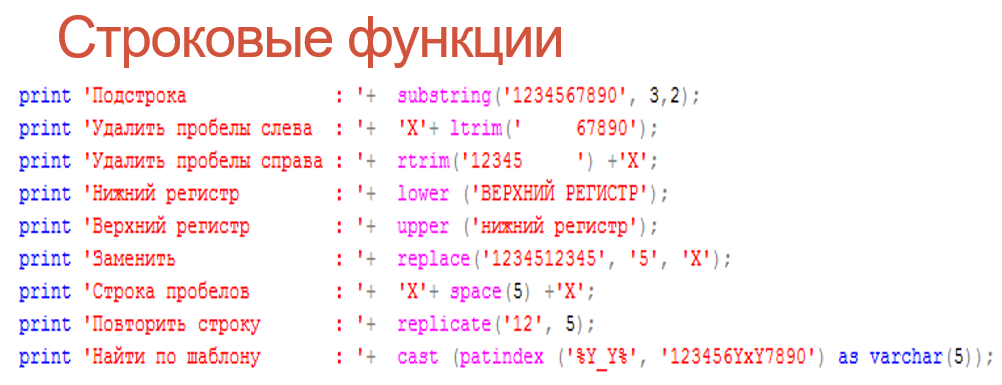
*Вывод данных* в T-SQL возможен двумя способами: оператором SELECT можно сформировать выходной результирующий набор и с помощью оператора PRINT можно вывести строку в стандартный выходной поток.

Если одновременно выводятся данные, сформированные оператором SELECT и данные, сформированные оператором PRINT, то посмотреть последние можно на вкладке Messages.

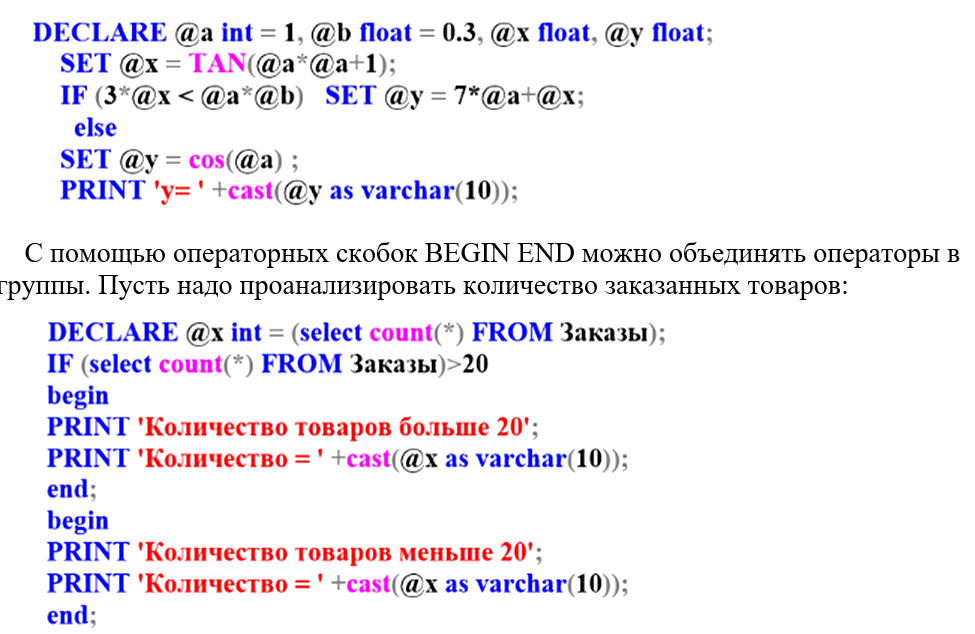


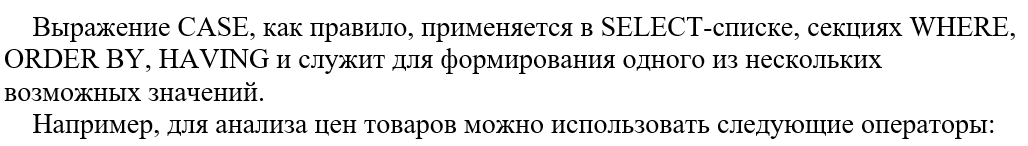


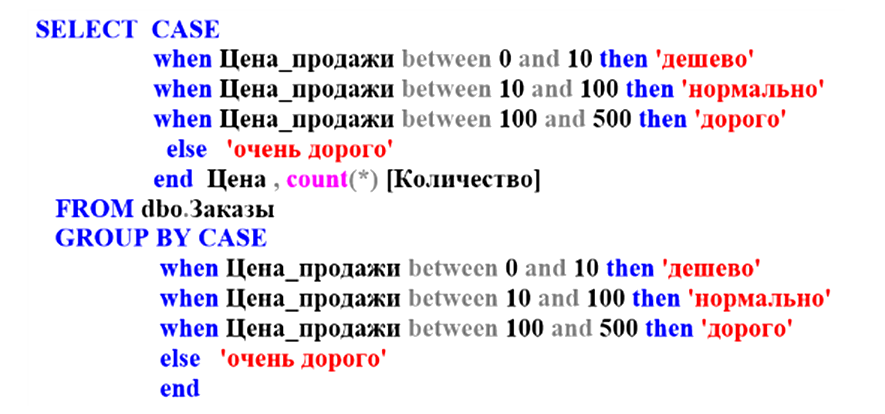


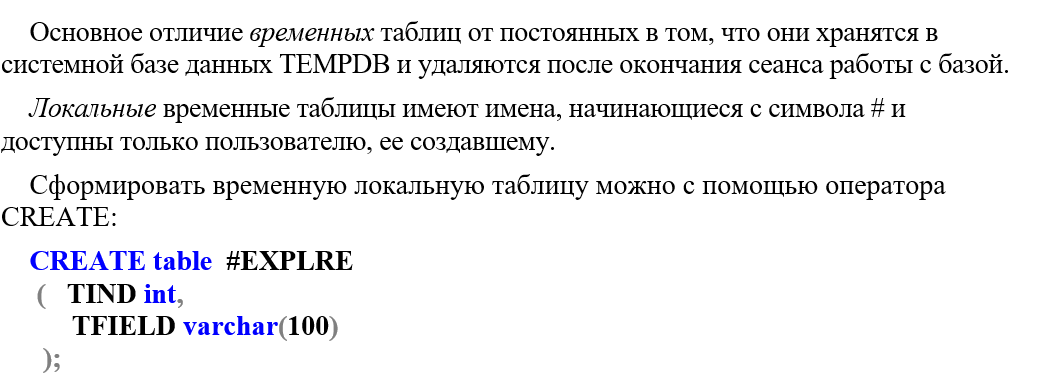


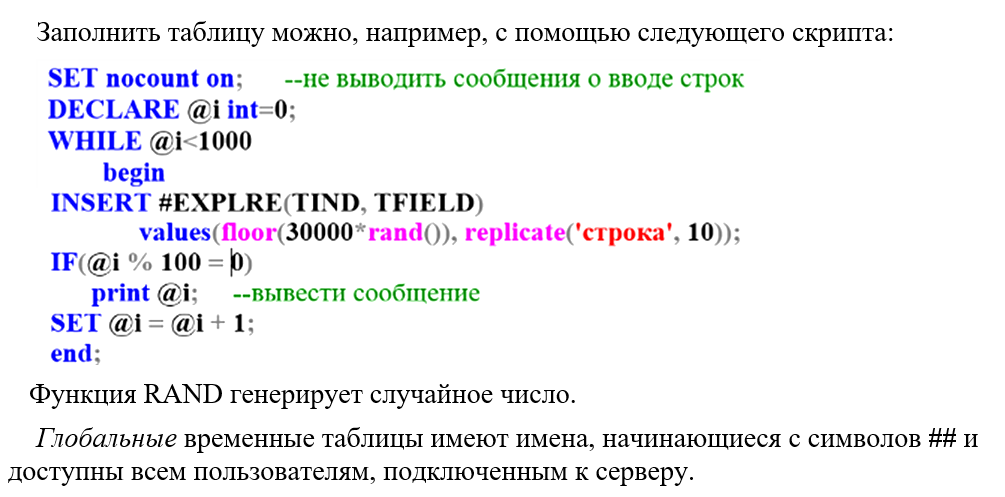
Функция CONVERT также используется для преобразования типов. Третий параметр этой функции применяется при преобразованиях, использующих типы данных для хранения даты и времени, и задает стиль представления этих данных.

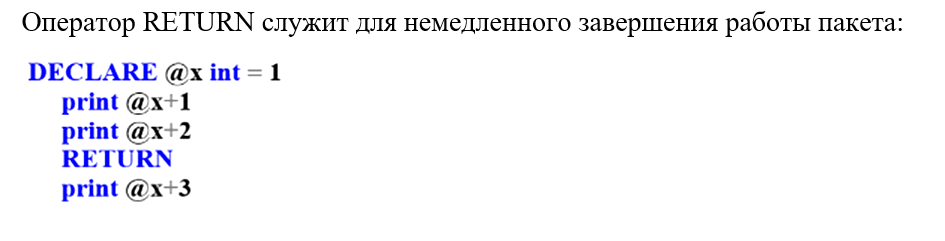


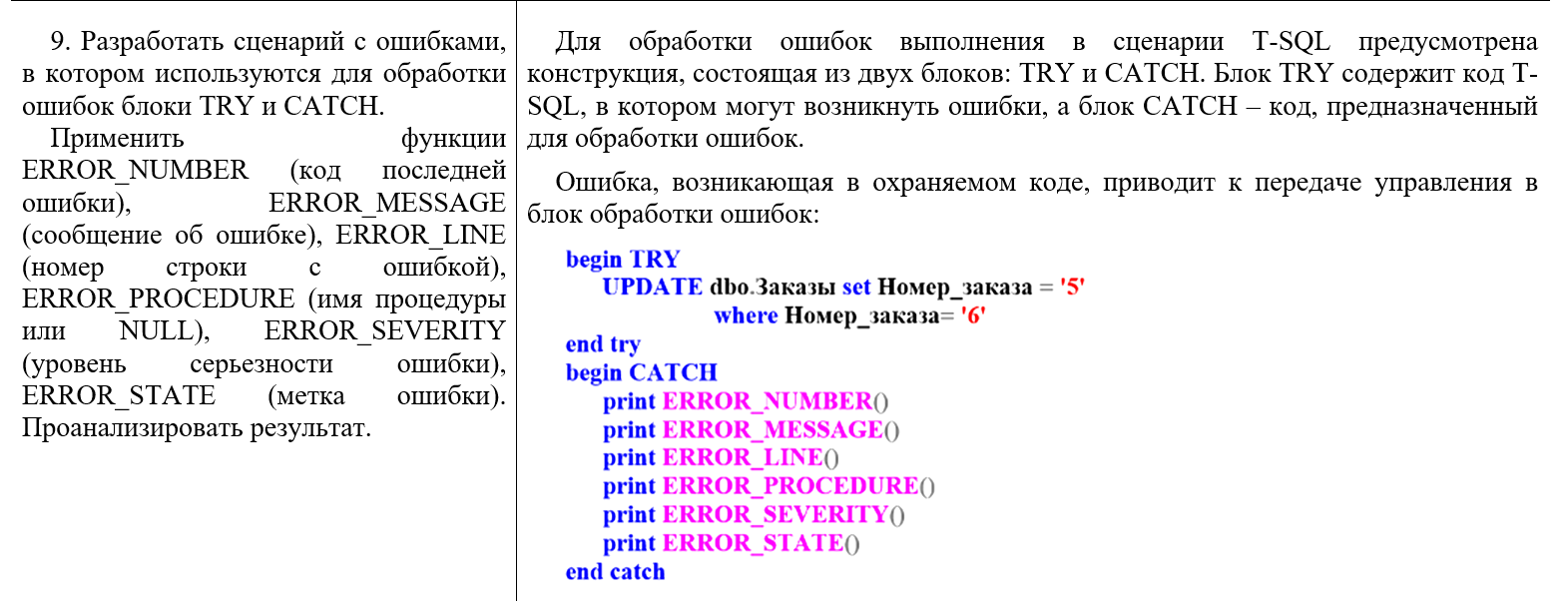






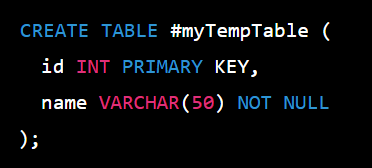






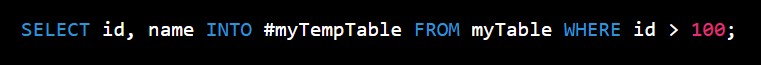
Временные таблицы в MS SQL Server - это специальный тип таблиц, которые создаются и используются в рамках текущей сессии или подключения к базе данных. Они могут быть локальными или глобальными, и в этом ответе я расскажу про временные локальные таблицы.

Временные локальные таблицы в MS SQL Server создаются с помощью ключевого слова CREATE TABLE #tableName или SELECT INTO #tableName, где #tableName - имя временной таблицы, которое начинается с символа "#". Например:



В этом примере мы создаем временную локальную таблицу #myTempTable с двумя столбцами: id типа INT и name типа VARCHAR(50).

Также мы можем создать временную локальную таблицу с помощью оператора SELECT INTO, который создает таблицу на основе результата запроса. Например:

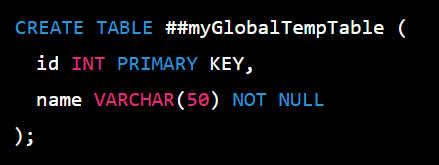


В этом примере мы создаем временную локальную таблицу #myTempTable на основе результата запроса, который выбирает столбцы id и name из таблицы myTable, где id больше 100.

Временные таблицы могут использоваться для хранения и обработки временных данных в рамках текущей сессии или подключения к базе данных, например, для временного хранения промежуточных результатов запросов или для временного хранения данных при выполнении сложных вычислений. Они автоматически удаляются при завершении сессии или подключения, или явно удаляются оператором DROP TABLE.

В MS SQL Server существует еще один тип временных таблиц - глобальные временные таблицы. Они, в отличие от локальных временных таблиц, доступны для всех сеансов подключения к базе данных, а не только для текущей сессии.

Глобальные временные таблицы создаются с помощью ключевого слова CREATE TABLE ##tableName, где ##tableName - имя глобальной временной таблицы, начинающееся с символов "##". Например:



В этом примере мы создаем глобальную временную таблицу ##myGlobalTempTable с двумя столбцами: id типа INT и name типа VARCHAR(50).

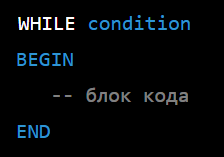
Глобальные временные таблицы используются в тех случаях, когда необходимо обеспечить доступ к временным данным из разных сеансов подключения к базе данных. Они могут использоваться для временного хранения данных, которые должны быть доступны из разных сеансов, или для обмена данными между различными приложениями или компонентами, которые работают с базой данных.

В глобальных временных таблицах также действуют ограничения на использование индексов, ключей и ограничений целостности, а также на доступ к данным из разных сеансов подключения к базе данных. Кроме того, глобальные временные таблицы автоматически удаляются при завершении всех сеансов, которые используют эту таблицу.

В MS SQL Server существуют два основных типа циклов: цикл WHILE и цикл FOR. Оба типа циклов позволяют повторять блок кода, пока выполняется определенное условие

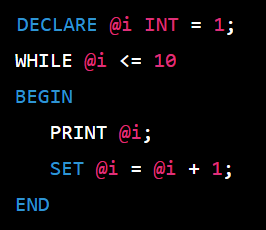
Цикл WHILE:

Цикл WHILE позволяет повторять блок кода до тех пор, пока условие истинно. Синтаксис цикла WHILE выглядит следующим образом:



где condition - логическое выражение, определяющее условие, при котором будет продолжаться выполнение цикла.

Пример использования цикла WHILE:



В этом примере мы объявляем переменную @i со значением 1 и запускаем цикл WHILE, который будет повторять блок кода, пока значение переменной @i меньше или равно 10. Внутри цикла мы выводим значение переменной @i на экран с помощью команды PRINT и увеличиваем значение переменной @i на 1 с помощью выражения SET @i = @i + 1.