**Введение в программирование на языке КОНЦЕПТ**

Редакция от 6 августа 2014г.

[www.gendoc.ru](http://www.gendoc.ru)

Оглавление

[Введение 1](#_Toc395089840)

[Области применения КОНЦЕПТ 3](#_Toc395089841)

[Команды 4](#_Toc395089842)

[Переменные 5](#_Toc395089843)

[Аргументы команд 7](#_Toc395089844)

[Списки 9](#_Toc395089845)

[Несколько примеров программ КОНЦЕПТ 10](#_Toc395089846)

[Заключение 11](#_Toc395089847)

# Введение

*Язык программирования КОНЦЕПТ* – это язык для написания компьютерных программ на русском языке.

Написанные на языке КОНЦЕПТ программы могут быть выполнены в специально разработанной и свободно распространяемой программе, называемой *Интерпретатор КОНЦЕПТ*. В настоящее время существует реализация Интерпретатора КОНЦЕПТ только для операционной системы *Windows*. На текущий момент актуальна версия

Для того чтобы начать использовать КОНЦЕПТ на своем компьютере, работающем под управлением Windows, достаточно загрузить с сайта [www.gendoc.ru](http://www.gendoc.ru) программу интерпретатора (файл *Concept.exe*), сохранить ее в любом каталоге компьютера и затем запустить файл *Concept.exe* на выполнение. На данный момент актуальной версией программы Интерпретатора КОНЦЕПТ является версия 1.7 от 10.04.2014.

При первом запуске Интерпретатора КОНЦЕПТ появляется окно, показанное на скриншоте ниже.



Это окно содержит следующие основные области:

* главное меню (в верхней части окна);
* панель для быстрого набора команд и специальных символов, используемых в языке программирования КОНЦЕПТ (в левой части окна);
* область кода, предназначенную для ввода команд или загрузки программы КОНЦЕПТ (в правой верхней части окна);
* область результата (в правой нижней части окна).

Писать программы КОНЦЕПТ можно непосредственно в области кода интерпретатора, либо в каком-либо текстовом редакторе, например в блокноте Windows. Программа КОНЦЕПТ может быть сохранена в простом текстовом файле с расширением *.txt*

Чтобы выполнить программу КОНЦЕПТ, текст которой был предварительно введен в область кода, либо загружен в область кода из текстового файла (используя последовательность пунктов главного меню: *Код => Загрузить…*), достаточно нажать кнопку с “галочкой”, расположенную непосредственно под строкой главного меню.

Ниже на скриншоте приведен исходный код и результат работы простейшей программы КОНЦЕПТ.



Набранный в области кода текст программы КОНЦЕПТ (но не загруженный из файла!) автоматически сохраняется в файле *\_Noname\_.txt* при завершении работы с Интерпретатором КОНЦЕПТ. При следующем запуске интерпретатора в область кода автоматически будет загружен ранее сохраненный текст из файла *\_Noname\_.txt*

# Области применения КОНЦЕПТ

КОНЦЕПТ предназначен для *непромышленного* программирования. Это не значит, что он является “сырым” или “неполноценным” продуктом. Просто задачи и, как следствие, свойства и сложность языка и среды разработки для промышленного и непромышленного программирования отличаются.

К непромышленному программированию относятся (и это достаточно большие классы задач):

* обучение алгоритмизации и программированию;
* разработка программ для научных исследований и экспериментов;
* "бытовое" программирование;
* быстрая разработка прототипов программ;
* моделирование бизнес-логики программ и процессов;
* генерация документов по шаблонам;
* разработка программ для одноразовых задач (например, преобразование данных).

Для непромышленного программирования не так важны высокая производительность, многопользовательский режим работы, поддержка сложных промышленных стандартов и ряд других характеристик. Возможные ошибки в непромышленных программах не приводят к таким серьезным последствиям, к каким могут приводить ошибки в промышленных программах.

В противоположность промышленному программированию, язык и среда разработки для непромышленных программ имеют следующие преимущества:

* простота установки и освоения среды для разработки программ;
* простота изучения языка программирования;
* высокий уровень абстракции встроенных в язык понятий и команд (это позволяет писать более короткие программы);
* ориентация на программирование в стиле “разворачивающейся спирали”: написал-запустил-сразу проверил-дописал-запустил-снова проверил…;
* поддержка творческого процесса программирования (использование разных стилей программирования, символьные вычисления, параметризация и самомодификация кода программы и т.д.);
* необязательность объявления переменных и указания их типов.

*Порог входа* в сферу непромышленного программирования, существенно ниже, чем в сферу промышленного.

Чтобы использовать КОНЦЕПТ, не нужно быть опытным или профессиональным программистом. Освоение КОНЦЕПТ вполне по силам и может помочь в решении задач следующим категориям пользователей:

* эксперты, аналитики и специалисты в различных предметных областях;
* менеджеры среднего и нижнего звена;
* научные работники;
* преподаватели, студенты и школьники;
* программисты и WEB-разработчики, использующие КОНЦЕПТ в качестве дополнительного инструмента.

# Команды

*Программа* КОНЦЕПТ – это последовательность *коман*д.

Каждая команда записывается с новой строки и начинается с *названия* команды, за которым могут следовать *аргументы (параметры)* команды, разделяемые пробелами.

Название команды состоит из *одного или двух ключевых слов*. Количество и семантика (смысл) аргументов команды определяется ее названием. В качестве аргументов команд могут выступать числа, строки символов, значения переменных, списки и др.

Например, команда

показать сообщение Привет!

имеет название показать сообщение и один аргумент Привет!

Команда

вычислить + ‘Сумма первых пяти чисел’ 1 2 3 4 5

имеет название вычислить +и шесть аргументов: Сумма первых пяти чисел*,* 1*,* 2*,* 3*,* 4*,* 5*.*

Команды делятся на *простые* и *составные*. Составных команд в КОНЦЕПТ всего несколько и они, в основном, предназначены для управления порядком вычислений (ветвление, цикл, определение и вызов функции). Ниже приведен пример составной команды ветвления:

Если [А] > 1

Печать “Значение переменной А больше 1.”

Иначе

Печать “Значение переменной А меньше либо равно 1.”

Конец

Ключевые слова в названии команды могут быть записаны в любом регистре. Ниже приведены примеры эквивалентной записи для команды с названием строка копировать:

строка копировать подстрока “Привет, Мир!” 1 6

СТРОКА КОПИРОВАТЬ подстрока “Привет, Мир!” 1 6

Строка копировать подстрока “Привет, Мир!” 1 6

СтрокА КопироватЬ подстрока “Привет, Мир!” 1 6

Во всех четырех указанных выше вариантах записи команды, в результате ее выполнения переменной с именем подстрока будет присвоено значение *Привет*.

Строки программы, начинающиеся с символа “\*” (звездочка) являются комментариями и игнорируются интерпретатором в процессе выполнения программы.

Строки, начинающиеся с символа “>” (знак больше) являются

# Переменные

В процессе своей работы программы, если только они не слишком примитивны, используют *переменные*. Переменные используются для временного хранения исходных данных и промежуточных вычислений программы. Все переменные КОНЦЕПТ хранятся в *рабочей памяти* интерпретатора и существуют только в процессе выполнения программы. В КОНЦЕПТ существуют *простые переменные*, *структурные переменные* (называемые также *сущностям*и) *и встроенные переменные*.

Под *простой переменной* понимается пара *имя переменной – значение переменной*.

Именем переменной в КОНЦЕПТ может быть любая непустая последовательность символов. Например, допустимыми именами переменных в КОНЦЕПТ, являются:

Взлетная масса самолета

Имя\_героя

Ячейка-игрового-поля[2,5]

Персона;Имя

Объект->Местоположение->Долгота

1 января 2014 года

123

+/-

(1+1+7)

Значением простой переменной может являться любая строка символов, включая пустую строку.

Под *структурной переменной* в КОНЦЕПТ понимается множество простых переменных рабочей памяти, имена которых имеют общее начало (префикс) до символа *“*;*”* (точка с запятой).

Например, четыре простые переменные:

Персона;Фамилия

Персона;Имя

Персона;Отчество

Персона;Дата рождения

Образуют структурную переменную с именем Персона и полями Фамилия, Имя, Отчество и Дата рождения.

Понятие структурной переменной введено в КОНЦЕПТ для того, чтобы можно было манипулировать с множеством связанных по смыслу переменных как с единым объектом.

Чтобы использовать в тексте программы КОНЦЕПТ значение переменной, нужно ее имя заключить в квадратные скобки [].

Например:

показать сообщение [Персона;Фамилия]

*Рабочую память*, в которой в момент выполнения программы хранятся все ее переменные, можно представить себе в виде таблицы из двух столбцов с заголовками “Имя переменной” и “Значение переменной”. В строке таблицы в первом столбце записывается имя переменной, а во втором столбце соответствующее этой переменной ее текущее значение. Все строки таблицы упорядочены (отсортированы) в алфавитном порядке по именам переменных. Ниже приведен возможный пример для состояния рабочей памяти некоторой вымышленной программы:

|  |  |
| --- | --- |
| **Имя переменной** | **Значение переменной** |
| Выбранный металл | золото |
| Сколько вешать в граммах? | 51 грамм |
| Остатки на складе;золото | 459 грамм |
| Остатки на складе;серебро | 1,7 кг |
| Остатки на складе;медь | 3 тонны |
| Покупатель | ООО “Синее море” |

В Интерпретаторе КОНЦЕПТ реализована обработка ряда *встроенных переменных*. Встроенные переменные начинаются со знака *“*$*”* (доллар). Например, значением встроенной переменной $Сегодня является текущая дата, а значение встроенной переменной $Время – текущее время.

Чтобы вывести в область результата интерпретатора текущие дату и время можно воспользоваться командой:

печать “Сегодня и сейчас” [$Сегодня] [$Время]

В окне результата появится что-то вроде:

*Сегодня и сейчас:*

*06.08.2014*

*07:12:20*

Полный перечень встроенных в Интерпретатор КОНЦЕПТ переменных можно посмотреть по ссылке <http://www.gendoc.ru/HelpCommands.htm#vars>

Переменные в КОНЦЕПТ специальным образом объявлять не требуется. Переменная создается в процессе выполнения программы в момент ее первого упоминания в качестве аргумента команды. Если на момент первого обращения к значению переменной значение переменной не было определено, то интерпретатор выдаст диалоговое окно с запросом к пользователю для ввода значения соответствующей переменной.

Имена переменных чувствительны к регистру. Например, переменные с именами ВЫСОТА, Высота, высота – это три разные переменные, которым будет соответствовать три различных записи в таблице рабочей памяти интерпретатора.

# Аргументы команд

Способ выполнения команд Интерпретатором КОНЦЕПТ является ключом к пониманию его работы и основным принципам, положенным в основу языка программирования КОНЦЕПТ.

Как было указано ранее, каждая команда начинается с названиякоманды, за которым могут следовать аргументы (параметры) команды, разделяемые пробелами.

После того, как интерпретатор распознает называние команды, он сначала *последовательно считывает и вычисляет все ее аргументы* и только потом выполняет саму команду. Под вычислением аргумента команды здесь понимается переход от (синтаксической) формы записи аргумента в строке команды к соответствующему этому аргументу значению.

По форме синтаксической записи аргументом команды может быть (и только!):

* строка символов;
* значение переменной (подстановка);
* строка символов с подстановками;
* вложенная команда.

В зависимости от названия команды и места расположения аргумента в ней, строка символов, например, может быть интерпретирована как число или список.

Ниже приведены правила вычисления значений аргументов.

Если аргументом является строка символов, заключенная в ‘’ (одинарные кавычки), то значением является собственно строка символов, заключенная в эти кавычки (т.е. кавычки просто отбрасываются, а то, что было внутри них и есть значение).

Если аргументом является последовательность символов, заключенная в [] (квадратные скобки), то символы внутри квадратных скобок интерпретируются как имя переменной рабочей памяти, а значением такого аргумента является значение соответствующей переменной. Такая запись также называется в КОНЦЕПТ *подстановкой*. Обрабатывая аргументы такого вида, интерпретатор сначала как бы подставляет в команду вместо этих аргументов значения соответствующих им переменных, и только потом выполняет саму команду, используя подставленные значения вместо аргументов.

Если аргументом является строка символов, заключенная в “” (двойные кавычки), то значением является строка символов, в которой выполнены все встречающиеся внутри нее подстановки, заключенные в символы [].

Если аргументом является команда, заключенная в () (круглые скобки), то значением является результат выполнения данной команды.

Если аргументом является строка символов, которая начинается с символа, отличного от символов ‘,”,[,(.$, то значенем такого аргумента является последовательнось этих символов до первого пробела.

Вложенность подстановок и команд может быть произвольной.

Ниже приведены примеры записи аргументов и соответствующие им значения (предполагается, значением переменной А является *1*, значением переменной Б является *2*, значением переменной Ячейка12 является *9999*):

|  |  |  |
| --- | --- | --- |
| **Запись аргумента в команде** | **Вид аргумента** | **Значение аргумента** |
| ***1*** | Строка символов | 1 |
| ***яблоко*** | Строка символов | яблоко |
| ***яблоко;груша;слива*** | Строка символов | яблоко;груша;слива |
| ***1.25e-5*** | Строка символов | 1.25e-5 |
| ***‘1.25e-5’*** | Строка символов | 1.25e-5 |
| ***“1.25e-5”*** | Строка символов | 1.25e-5 |
| ***‘город Москва’*** | Строка символов | город Москва |
| ***1+1*** | Строка символов | 1+1 |
| ***‘1+1’*** | Строка символов | 1+1 |
| ***‘[А]’*** | Строка символов | [А] |
| ***[А]*** | Подстановка | 1 |
| ***[Б]*** | Подстановка | 2 |
| ***[Ячейка[А][B]]*** | Подстановка вложенная | 9999 (сначала формируется имя переменной Ячейка12, затем берется ее значение 9999) |
| ***“Сначала [А], а потом [Б]”*** | Строка символов с подстановками | Сначала 1, а потом 2 |
| ***“В полдень ровно [А][Б] часов.”*** | Строка символов с подстановками | В полдень ровно 12 часов. |
| ***(строка копировать $результат ‘Привет, Мир!’ 1 6 )*** | Вложенная команда | Привет |
|  ***(вычислить + $результат 3 3 )*** | Вложенная команда | 6 |
| ***(строка копировать $результат ‘Привет, Мир!’ 1 (вычислить + $результат 3 3 ) )*** | Вложенная команда | Привет |

Если все аргументы команды записывать в одну строку неудобно, то для переноса аргументов на следующую строку нужно использовать символ “\” (обратный слеш). Например (предполагается, что переменная с именем пробел имеет значение равное одному символу пробела):

строка сцепить новое\_предложение Я [пробел] сегодня \

[пробел] вечером [пробел] \

пойду [пробел] в [пробел] кино

Результатом выполнения данной команды будет присвоение значения *Я сегодня вечером пойду в кино* переменной с именем новое\_предложение.

При использовании вложенной команды в качестве аргумента, *перед закрывающей скобкой необходимо ставить хотя бы один пробел*. Иначе закрывающая скобка будет восприниматься интерпретатором как часть ключевого слова или часть последнего аргумента команды.

Например, если не поставить пробел перед закрывающей скобкой в команде ниже

вычислить + сумма 1 2 (вычислить + $результат 1 2)

то вместо присвоения переменной сумма значения *6*, интерпретатор выдаст ошибку, т.к. воспримет 2) одним аргументом – строкой из двух символов.

# Списки

Списки являются очень важными элементами языка КОНЦЕПТ, т.к. с их помощью моделируются такие важные структуры данных как записи, множества, деревья, графы, отношения (факты) и др.

Строка символов в КОНЦЕПТ можно может быть интерпретирована как список. Элементы списка разделяются символом “;” (точка с запятой). Для задания вложенных списков используется пара {} (фигурные скобки), в которую заключаются элементы вложенного списка. Пустой список обозначается как пара фигурных скобок {}, которые внутри себя ничего не содержат.

Ниже в таблице приведены примеры списков.

|  |  |
| --- | --- |
| **Пример записи списка** | **Интерпретация списка** |
| 1 | Список из одного элемента |
| 1;2;3 | Список из трех элементов |
| столица;Россия;Москва | Список из трех элементов (моделирование отношения, факта) |
| 1;{2;3};{4;5};6 | Список из 4-х элементов, два из которых являются вложенными списками |
| {} | Пустой список |

# Несколько примеров программ КОНЦЕПТ

Ниже приведено несколько простых примеров программ КОНЦЕПТ, взятых с сайта [www.gendoc.ru](http://www.gendoc.ru)

Пример 1.Вычисление факториала

*\* Вычисление факториала натурального числа*

*\* Программа КОНЦЕПТ, 17.09.2010, www.gendoc.ru*

**функция** Факториал Число

 **присвоить** Индекс 2 Результат 1

 **пока** **[Индекс]** <= **[Число]**

 вычислить \* Результат **[Результат]** **[Индекс]**

 **увеличить** Индекс

 **цикл**

 печать "Факториал **[Число]**! = **[Результат]**."

**возврат**

Факториал 5

Факториал 10

Факториал 20

Пример 2.Генерация писем-приглашений в формате HTML

*\* Генерация писем-приглашений по списку персон*

*\* Программа КОНЦЕПТ, 08.12.2010-24.07.2012, www.gendoc.ru*

*\* Процесс разработки этой программы подробно описан на http://www.gendoc.ru/ConceptIdea.htm*

*\* Запрос значений переменных*

**присвоить** ПРИГЛАСИТЬ 'Иван Иванович;Пер Петрович;Анна Александровна;Дмитрий Дмитриевич'

ввести меню1 'Название мероприятия' 'Какое мероприятие?' 'на встречу выпускников;на соревнование по легкой атлетике;на званый ужин'

ввести переменная 'Дата проведения мероприятия' 'Когда?' **[$Сегодня]**

ввести меню1 'Время начала' 'Время?' '10:00;12:00;14:00;17:00;19:00'

ввести меню1 'Место проведения мероприятия' 'Где?' 'в ресторане "Вечерний";на стадионе "Метеор";у меня дома'

**присвоить** Подпись 'Иван Петров'

*\* Логическая обработка*

**присвоить** употребление\_который который

**выбрать** **[Название мероприятия]**

**вариант** 'на встречу выпускников'

 **присвоить** употребление\_который которая

**вариант** 'на соревнование по легкой атлетике'

 **присвоить** употребление\_который которое

**конецВыбора**

*\* Перенаправить вывод строк в файл*

**присвоить** ВРЕМЕННЫЙ\_HTML "**[$ТекущийКаталог]**tmp.htm"

файл писать **[ВРЕМЕННЫЙ\_HTML]**

*\* Начало HTML*

**><!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN">**

**><html>**

**><head>**

**><meta http-equiv="Content-Type" content="text/html; charset=windows-1251">**

**><title>Приглашения</title>**

**></head>**

**><body>**

*\* Генерация приглашений по шаблону в цикле*

**для** **[ПРИГЛАСИТЬ]**

*\* Шаблон текста*

**><table><tr>**

**><td><img src="http://www.gendoc.ru/Examples/flower.png"></td>**

**><td><h2 align="center" fontcolor="red">Уважаемый [$ОбъектЦикла]!</h2>**

**><p>Приглашаем Вас [Название мероприятия],**

**>[употребление\_который] состоится <b><u>[Дата проведения мероприятия] в [Время начала]**

**>[Место проведения мероприятия]</u></b>.**

**><br>Приходите обязательно!</p>**

**><i>С уважением,**

**><br>[Подпись]</i>**

**></td></tr></table>**

**><hr>**

**следующий**

*\* Конец HTML*

**></body>**

**></html>**

*\* Показ результата*

файл конецЗаписи

показать html **[ВРЕМЕННЫЙ\_HTML]**

# Заключение

Лучший способ освоить КОНЦЕПТ – это начать писать программы на нем. Несколько экспериментов с командами, аргументами и переменными позволит вам “прочувствовать” гибкость этого языка и подобрать для себя наиболее подходящий стиль программирования.

Актуальную версию Интерпретатора КОНЦЕПТ можно загрузить с сайта [www.gendoc.ru](http://www.gendoc.ru). На этом же сайте размещается актуальная справочная информация и примеры программ КОНЦЕПТ.

Справочник всех команд языка программирования КОНЦЕПТ и примеры их использования доступны по ссылке <http://www.gendoc.ru/HelpCommands.htm>

С примерами исходных текстов программ КОНЦЕПТ и результатами их работы можно ознакомиться по ссылке <http://www.gendoc.ru/ConceptExamples1.htm>

Загружать и выполнять демонстрационные примеры программ КОНЦЕПТ из Интернет можно непосредственно из программы Интерпретатора КОНЦЕПТ (Concept.exe), воспользовавшись последовательностью пунктов главного меню: *Код => Загрузить и выполнить пример с* [*www.gendoc.ru*](http://www.gendoc.ru)

***Успехов вам в освоении и использовании КОНЦЕПТ!!!***