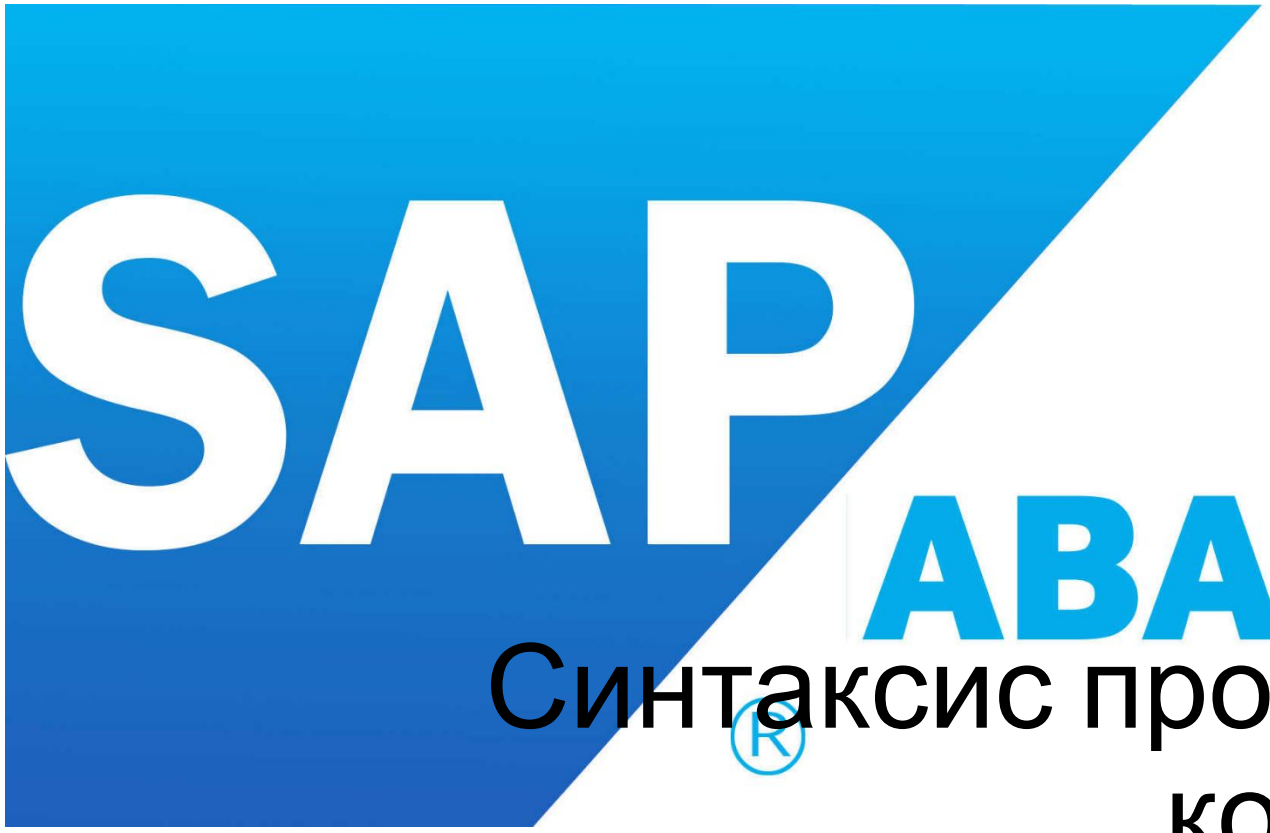

ABAP®

Синтаксис программы, операторы, комментарии, техника программирования. Основные принципы программирования.

Основными возможностями языка ABAP/4 являются:

- Разновидность самодокументирующегося синтаксиса
- Наличие встроенных типов данных
- Наличие встроенных интерфейсов взаимодействия с базой данных и удаленного доступа
- Встроенная поддержка отчетов, средств создания интерфейсов пользователя и OLTP программирования
- Система событий для написания отчетов и выборки данных
- Наличие удобный, адаптированных к БД структур данных с заложенными алгоритмами оптимизации доступа (внутренние таблицы)
- Средства обработки больших объемов данных
- Встроенная поддержка динамического программирования
- Поддержка как устаревших операторов, так и современного процедурно-ориентированного и объектно-ориентированного стиля программирования

Функции объектно-ориентированного программирования включают в себя поддержку:

- Инкапсуляции (классы и интерфейсы)
- Разделения областей видимости компонентов классов
- Наследования и вложенных интерфейсов
- Полиморфизма и позднего связывания
- Обработки событий на основе publish-subscribe модели

- Язык ABAP/4 является разновидностью интерпретатора. Компилятор ABAP/4 генерирует промежуточный код (byte-код), который потом выполняется системой.
Язык ABAP/4 является платформенно-независимым, и может использоваться с различными комбинациями БД, и ОС.

Структура и виды программ, среда разработки

Внутри системы R/3 ABAP/4 имеет собственную среду разработки. Среда разработки ABAP/4 состоит из следующих основных компонентов и прикладных приложений в системе R/3:

- Редактор ABAP/4 (se38)
- Отладчик (se38)
- Словарь данных (se11)
- Построитель функций (se37)
- Построитель классов (se24)
- Средства контроля времени выполнения и производительности (se38)
- Расширенная синтаксическая проверка программ (se38)
- Редактор экранов (se51)
- Редактор меню (se41)
- Навигатор по объектам (se80)
- Информационная система репозитария объектов
- Организатор переносов (se09)

В среде разработки ABAP/4 также поддерживаются следующие механизмы:

- Интерфейс Open SQL
- Интерфейс работы с файлами
- OLTP программирование (распределенные буферы, распределенная обработка блокировок, сложные объекты блокирования, распределение ресурсов рабочих процессов, асинхронная обработка обновлений)
- Система авторизации и доступа к объектам
- Интерфейсы взаимодействия с внешними системами (DCOM/CORBA, RFC, OLE, CPI/C)
- Пакетный ввод данных
- Поддержка различных кодировок и форматов номеров
- Локализация (привязка программно-аппаратных средств к условиям и стандартам страны пользователя)

Схема выполнения блоков обработки

Существуют следующие типы ABAP программ:

- Тип 1
- Тип M
- Тип F
- Тип K
- Тип J
- Тип S
- Тип I

Концепция синтаксиса языка ABAP/4

1. Любое выражение должно заканчиваться точкой.

2. Двоеточие после ключевого слова обозначает повторение выражений, приведенных после двоеточия и перечисленных через запятую для данного ключевого слова.

Код:

```
WRITE: 'Hello World',
    ' - this is my first programm'.
```

Равен последовательности:

```
WRITE 'Hello World'.
WRITE ' - this is my first programm'.
```

3. Ключевые слова и переменные регистронезависимые, т.е. код:

Код:

```
DATA gv_value TYPE i.
WRITE gv_value.
```

Идентичен коду:

```
data GV_VALUE type I.
wrITE gv_Value.
```

4. Вызов функциональных модулей производится только в верхнем регистре: Т.е. код:

```
...
CALL FUNCTION 'ztest_func'.
...
```

Работать не будет, а правильно:

```
...
CALL FUNCTION 'ZTEST_FUNC'.
...
```

Определение данных в языке ABAP/4

Элементарные типы данных

В ABAP/4 используются следующие элементарные типы:

- Characters. Поддерживаются два символьных типа:
  C (собственно символьный) и N (текст, состоящий из цифр).
- Numbers. Поддерживается три цифровых типа:
  I (целые числа), P (упакованные числа) и F (числа с плавающей запятой).
- Date. Поддерживается один тип даты:
  D (дата).
- Time. Поддерживается один тип для задания времени: T (время).
- Hexadecimal.
  Поддерживается один шестнадцатеричный тип: X (шестнадцатеричный).

```
DATA:

name(25)  TYPE C,

z_code(5) TYPE N,

counter   TYPE I VALUE 1



TYPES
t_flag  TYPE C.
DATA add_flag  TYPE t_flag.
```

Для всех типов в качестве значений можно задавать константы:

```
CONSTANTS:
company_name(3) TYPE C,
max_counter     TYPE I VALUE 9999.
```

Константы используются для определения начальных значений:

```
DATA:
counter TYPE I VALUE max_counter.
```

Сложные типы

Кроме полей ABAP/4 поддерживает специальные конструкции для сложных (или составных) объектов данных: записи и внутренние таблицы.

Структуры содержат фиксированное число объектов данных (компонентов структуры), определяемых с помощью ключевых слов DATA BEGIN OF и DATA END OF. Можно определить структуру со следующими полями.

```
DATA:
BEGIN OF customer,
      id(8)    TYPE n,
      name(25),
      telephone(12),
END OF customer.
```

После того как структура определена, можно работать и с отдельными компонентами и со всей структурой.

Пример:

```
DATA vendor LIKE customer.
    customer-id = 87654321.
    customer-name = Green.
    customer-telephone = 211-22-34.
MOVE customer  TO vendor.
```

```
TYPES: BEGIN OF address,
       city(25),
       street(30),
END OF address,
BEGIN OF person,
       name(25),
       address  type address,
END OF person.
DATA: receiver TYPE person.
DATA: receiver_tab LIKE receiver OCCURS 0
WITH HEADER LINE.
DATA: target  LIKE receiver OCCURS 0 WITH HEADER LINE.
```

Операторы языка ABAP/4

MOVE и COMPUTE.

Команда MOVE всегда копирует исходное поле в целевое.

MOVE: исходное TO целевое.

COMPUTE целевое = исходное.

Ключевое слово COMPUTE единственное, которое разрешается опускать в операторах языка.

Пример:

```
MOVE: receiver TO receiver_tab.
MOVE: receiver_tab TO target.
```

Операции с символьными строками.

Символьные строки объединяются с помощью команды CONCATENATE.

```
DATA: str1(3) VALUE 'Red',
str2(6) VALUE 'Yellou',
str3(6) VALUE 'Green',
str4(50).
CONCATENATE str1 str2 str3 INTO str4 SEPARATED BY ','.
```

Обратная операция разделения символьной строки осуществляется для произвольно выбранного разделителя.

Пример:

```
DATA:
list(40) VALUE 'Edison, Smith, Jon Green, Yang, Black',
name1(20), name2(20), name3(20), name4(20), name5(20).
SPLIT list AT ',' INTO name1  name2 name3  name4 name5.
```

Если одно из полей, в которое записывается результат, имеет недостаточную длину, все компоненты усекаются, и переменная sy-subrc получает ненулевое значение. Если число компонентов больше числа целевых полей, то информация теряется.

В качестве целевого объекта можно использовать внутреннюю таблицу.

```
DATA names LIKE name1 OCCURS 100.

SPLIT list AT ',' INTO TABLE names.
```

Сдвинуть символьную строку можно командой SHIFT.

```
SHIFT name1 BY 3 places.

SHIFT name2 RIGHT.

SHIFT  name3 UP TO 'Jon’.
```

Для замены определённых символов в строке используется оператор REPLACE, который замещает первую встретившуюся подстроку внутри строки.

Пример:

```
REPLACE E WITH Мав INTO list.
```

Поиск символьных строк в полях или внутренних таблицах осуществляется по команде SEARCH. Системное поле sy-fdpos содержит сдвиг найденной строки относительно начала.

Пример:

```
SEARCH list FOR Green.
IF sy-subrc NE 0.
WRITE ‘Not found’.
ENDIF.
```