ТЕОРИЯ И ПРАКТИКА ЭКОНОМИЧЕСКИХ ИНФОРМАЦИОННЫХ СИСТЕМ

Универсальные коллекции значений встроенного языка

Платформа 1С:Предприятие 8

Платформа позволяет разрабатывать информационные системы различного назначения. Относится к предметно-ориентированным средствам разработки.

Решения, построенные на основе Платформы принято называть **конфигурациями**. Конфигурации строятся из базовых объектов платформы — **объектов метаданных**.

Для реализации алгоритмов конфигурации, разрабатываемой в рамках Платформы, применяется **встроенный язык**.

Примитивные типы данных встроенного языка

- Число
- Строка
- Дата (дата, время, дата и время)
- Булево (Истина или Ложь)
- Неопределено (для определения отсутствующего значения)
- Null (для определения отсутствующего значения в таблицах базы данных)
- Тип (необходимо для представления и сравнения типов данных)

Конструкции встроенного языка Платформы - Присваивание

Динамическая типизация — переменная связывается с типом данных в момент присваивания значения, а не в момент объявления переменной.

Переменную можно объявить заранее, используя оператор *Перем*.

```
Переменная1 = Переменная2;
ПерСтрока = "2s8dgaaфффф";
ПерЧисло = 123;
ПерБулево = НЕ Истина;
ПерДата = "20080717";
ПерДата = Дата(2008, 07, 17);
ПерОбъект = Справочники.Товары
    .НайтиПоНаименованию("Товар1");
```

Конструкции встроенного языка Платформы - Условия

```
Если <логич. выражение 1>
    ИЛИ <логич. выражение 2>
    И НЕ <логич. выражение 3> Тогда
    ...
ИначеЕсли <логич. выражение 4> Тогда
    ...
ИначеЕсли <логич. выражение 5> Тогда
    ...
Иначе
    ...
КонецЕсли;
```

```
// сокращенная форма записи
?(<логич. выражение 1>, <действие истина>, <действие ложь>)

?(<логич. выражение 1>,
     ?(<логич. выражение 2>, <действие истина 2>, <действие ложь 2>),
     <действие ложь 1>)
```

Конструкции встроенного языка Платформы - Циклы

```
Для <переменная счетчик цикла> = <начальное значение>
    По <конечное значение> Цикл
    ...
    Если <переменная счетчик цикла> = 10 Тогда
        Прервать; // Прерывание цикла
    КонецЕсли;
    Если <переменная счетчик цикла> = 0 Тогда
        Продолжить; // Переход к следующей итерации
    КонецЕсли;
КонецЦикла;
```

```
Для Каждого <элемент коллекции> Из <коллекция> Цикл
    ...
КонецЦикла;

//////////////////////////////////////////////////////

Пока <логич. выражение> Цикл
    ...
КонецЦикла;
```

Конструкции встроенного языка Платформы – Процедуры и Функции

```
// Переменная1 передается по ссылке
// Переменная2 передается по значению
// Если Переменная3 не определена, используется значение
// по умолчанию (Ложь)
Процедура Тест(Переменная1, Знач Переменная2,
    Переменная3 = Ложь)
    ...
    Если НЕ Переменная3 Тогда
        Возврат; // Выход из процедуры
    КонецЕсли;
    ...
КонецПроцедуры;
```

```
Функция ФТест(Переменная1, Знач Переменная2,
    Переменная3 = Ложь)
    ...
    // Предварительное объявление локальной переменной
    Перем Количество;
    ...
    Возврат Количество;
КонецФункции;
```

Универсальные коллекции значений

- Массив
- Фиксированный массив
- Список значений
- Таблица значений
- Структура
- соответствие

Массив

Массив — структура данных, хранящая набор значений (элементов массива), идентифицируемых по индексу

К элементу массива можно обращаться по индексу через оператор [...]

Индекс первого элемента равен 0.

В массив можно загружать данные из других коллекций или выгружать данные из него в другие коллекции.

Создание нового массива

```
// Одномерный массив без элементов
Мас1 = Новый Массив();

// Одномерный массив из 10 элементов
// При создании массива было создано 10 элементов
// со значение Неопределено
Мас2 = Новый Массив(10);

// Двумерный массив (матрица 4 на 5)
Мас3 = Новый Массив(4,5);
```

Изменение массива

```
Мас1 = Новый Массив();
Мас1.Добавить("Иванов"); // Индекс = 0
Мас1.Добавить("Петров"); // Индекс = 1
Мас1.Добавить("Сидоров"); // Индекс = 2

// После вставки элемента Сидоров элементы Иванов и Петров будут сдвинуты ниже
Мас1.Вставить(0,"Сидоров");

// После удаления элемента Сидоров элементы Иванов и Петров сдвинутся обратно
Мас1.Удалить(0);
```

Методы массива

- ВГраница (UBound) - получает наибольший индекс элемента массива
- Количество(Count) - Получает количество элементов в массиве.
- Найти (Find) - Выполняет поиск элемента в массиве
- Очистить (Clear) - Удаляет все значения из массива.
- Получить (Get) - Получает значение по индексу. Работает аналогично оператору []
- Установить (Set) - Устанавливает значение по индексу. Работает аналогично оператору [].

Перебор элементов массива

```
//1-й вариант
Для Каждого Элемент Из Массив Цикл
    Сообщить(Элемент);
КонецЦикла;

//2-й вариант
Для Ит = 0 По Массив.ВГраница() Цикл
    Сообщить(Массив[Ит]);
КонецЦикла;

//3-й вариант
Для Ит = 0 По Массив.Количество() - 1 Цикл
    Сообщить(Массив.Получить(Ит));
КонецЦикла;
```

Фиксированный массив

Фиксированные массивы создаются на основе «обычных» массивов:

При использовании фиксированного массива:

- нельзя менять значения элементов, находящихся в массиве;
- нельзя добавлять новые элементы в массив;
- нельзя удалять имеющиеся элементы из массива

Список значений

Список значений — это аналог одномерного массива, который предназначен в основном для решения задачи выбора значения из предопределенного списка при разработке графического интерфейса пользователя.

Список значений можно представить в виде множества элементов с типом ЭлементСпискаЗначений

Свойства элементов списка значений

- Значение — хранимое значение.
- Представление — пользовательское представление значения, которое будет показано при выводе значения на экран. Если представление не указано, то оно формируется системой автоматически на основе значения.
- Пометка — используется для отметки некоторых значений из списка.
- Картинка — графическое изображение, связанное с данным значением

Методы списка значений

```
Список = Новый СписокЗначений();

// Удаление всех элементов
Список.Очистить();

// Добавление элемента Иванов (Индекс = 0) с указанием представления Иванов Иван
Список.Добавить("Иванов", "Иванов Иван");

// Добавление элемента Петров (Индекс = 1)  с указанием представления Петров Петр
// и установкой пометки в значение Истина
Список.Добавить("Петров", "Петров Петр", Истина);

// Удаление элемента Петров (Индекс = 1)
Список.Удалить(1);

// После вставки элемента Сидоров в начало списка элемент Иванов сдвигается вниз (Индекс = 1)
Список.Вставить(0, "Сидоров");

// Сдвиг элемента Иванов (Индекс = 1) на одну позицию вверх
Список.Сдвинуть(1, -1);
```

Перебор элементов списка значений

```
// 1-й вариант
Для Каждого Элемент Из Список Цикл
    Сообщить(Элемент.Значение);
КонецЦикла;

// 2-й вариант
Для Ит = 0 По Список.Количество() - 1 Цикл
    Сообщить(Список[Ит].Значение);
КонецЦикла;

// 3-й вариант
Для Ит = 0 По Список.Количество() - 1 Цикл
    Сообщить(Список.Получить(Ит).Значение);
КонецЦикла;

// Поиск элемента в списке значений
Элемент = Список.НайтиПоЗначению("Иванов");
Если Элемент <> Неопределено Тогда
    Сообщить("Элемент найден!");
КонецЕсли;
```

Таблица значений

Таблица значений состоит из строк и колонок.

В отличие от списка значений, таблица значений имеет структуру, которую определяет разработчик.

При работе со строками таблицы значений необходимо понимать, что строка — это объект типа СтрокаТаблицыЗначений.

Таблица значений является полностью динамическим объектом, т.е. можно манипулировать не только строками таблицы, добавляя и удаляя их, но и колонками.

Колонки таблицы значений

Прежде чем начать работу с таблицей значений, необходимо создать структуру колонок. Каждая колонка характеризуется следующими свойствами:

- Имя — идентификатор колонки (может содержать только алфавитные символы, цифры и знаки подчеркивания. Причем, начинаться имя колонки может только с буквы или символа подчеркивания);
- Заголовок — представление колонки в диалогах (может содержать произвольные символы);
- ТипЗначения — тип значения содержимого ячеек в этой колонке. Если тип не задан, в ячейке можно хранить значения произвольного типа;
- Ширина — ширина колонки в диалогах;

Создание колонок таблицы значений

```
Таблица = Новый ТаблицаЗначений();
Таблица.Колонки.Добавить("Наименование");
Таблица.Колонки.Добавить("Цена");
Таблица.Колонки.Добавить("Количество");
```

Методы колонок таблицы значений

Вставить()	Вставляет новую колонку в указанную позицию коллекции
Добавить()	Добавляет новую колонку в конец коллекции
Количество()	Возвращает количество колонок в коллекции
Найти()	Ищет колонку в коллекции по имени
Очистить()	Удаляет все колонки из коллекции
Сдвинуть()	Сдвигает колонку влево или вправо
Удалить()	Удаляет колонку из коллекции

Методы таблицы значений

```
// Добавление строки
НовСтрока = Таблица.Добавить();
НовСтрока.Ссылка = Ссылка;
НовСтрока.Наименование = "Товар 1";
НовСтрока.Цена = 14.1;
НовСтрока.Количество = 100;

// Добавление строки в начало таблицы значений
НовСтрока = Таблица.Вставить(0);
НовСтрока.Ссылка = Ссылка;
НовСтрока.Наименование = "Товар 2";
НовСтрока.Цена = 29.3;
НовСтрока.Количество = 54;

// Сдвиг 2-й строки на одну позицию вверх
Таблица.Сдвинуть(1, -1);

// Удаление 1-й строки
Таблица.Удалить(0);

// Удаление всех строк
Таблица.Очистить();
```

Перебор строк таблицы значений

```
// 1-й способ
Для Каждого Строка Из Таблица Цикл
    Сообщить(Строка.Наименование);
    Сообщить(Строка.Цена);
КонецЦикла;

// 2-й способ
Для Ит = 0 По Таблица.Количество() - 1 Цикл
    Сообщить(Таблица[Ит].Наименование);
    Сообщить(Таблица[Ит].Цена);
КонецЦикла;

// 3-й способ
Для Ит = 0 По Таблица.Количество() - 1 Цикл
    Строка = Таблица.Получить(Ит);
    Сообщить(Строка.Наименование);
    Сообщить(Строка.Цена);
КонецЦикла;
```

Поиск строк в таблице значений

```
// Поиск строки
// Метод Найти возвращает ссылку на строку таблицы значений
Строка = Таблица.Найти(14.1, "Цена");
Если Строка <> Неопределено Тогда
      Сообщить("Товар с такой ценой найден!"
            + Строка.Наименование);
КонецЕсли;

// Поиск нескольких строк
// Метод НайтиСтроки возвращает массив ссылок на строки таблицы значений
Отбор = Новый Структура("Количество", 100);
МассивСтрок = Таблица.НайтиСтроки(Отбор);
Сообщить("Найдено " + МассивСтрок.Количество()
      + " товаров с количеством = 100");
```

Структура и соответствие

Структура и соответствие являются динамическими коллекциями значений.

Каждый элемент такой коллекции представляет собой пару «ключ» и «значение».

Ключи структуры и соответствия уникальны, поэтому они однозначно идентифицируют элемент коллекции.

В структуре ключи могут иметь только строковый тип данных и должны подчиняться правилам именования переменных.

В соответствиях ключи могут иметь почти любой тип данных.

Структура

Структуры чаще используются для организации отборов, передачи списка параметров

```
// Создание новой структуры с ключами ДатаНач и ДатаКон
Параметры = Новый Структура("ДатаНач, ДатаКон");
// Установка значения для ключа ДатаНач
Параметры.ДатаНач = Дата(1900, 01, 01);
// Установка значения для ключа ДатаКон
Параметры.ДатаКон = ДобавитьМесяц(Параметры.ДатаНач, 1);
// Добавление элемента с указанием ключа и значения
Параметры.Вставить("СворачиватьИтоги", Истина);
СформироватьОтчет(Параметры);

//Перебор элементов структуры
Для Каждого Элемент Из Параметры Цикл
    Ключ = Элемент.Ключ;
    Значение = Элемент.Значение;
КонецЦикла;
```

Соответствие

```
// Создание нового соответствия
ПараметрыОбработки = Новый Соответствие;
// Вставка элемента соответствия
ПараметрыОбработки.Вставить("Дата начала", ДатаНач);
ПараметрыОбработки.Вставить("Дата конца", ДатаКон);
// Удаление элемента соответствия по ключу
ПараметрыОбработки.Удалить("Дата начала");
// Удаление всех элементов
ПараметрыОбработки.Очистить();

Для Каждого Элемент Из ПараметрыОбработки Цикл
    Сообщить(Элемент.Ключ + " - " + Элемент.Значение);
КонецЦикла;
```

Сортировки

Гномья сортировка - алгоритм похожий на сортировку вставками, но перед вставкой на нужное место происходит серия обменов, как в сортировке пузырьком.

«Это метод, которым садовый гном сортирует линию цветочных горшков. По существу он смотрит на текущий и предыдущий садовые горшки: если они в правильном порядке, он шагает на один горшок вперёд, иначе он меняет их местами и шагает на один горшок назад. Граничные условия: если нет предыдущего горшка, он шагает вперёд; если нет следующего горшка, он закончил»

Пример

Массив [4] [2] [7] [3]:

[4] [2] [7] [3] (начальное состояние: i == 0, j == 1);

[4] [2] [7] [3] (ничего не произошло, но сейчас i == 1, j == 2);

[4] [7] [2] [3] (обмен a[2] и a[1], сейчас i == 0, а j == 2 по-прежнему);

[7] [4] [2] [3] (обмен a[1] и a[0], сейчас i == 2, j == 3);

[7] [4] [3] [2] (обмен a[3] и a[2], сейчас i == 1, j == 3);

[7] [4] [3] [2] (ничего не произошло, но сейчас i == 3, j == 4);

цикл закончился, т. к. i не < 3.