Модула-2

Тип запись

Не должен быть пустым (не содержащим полей), поскольку его нельзя расширить. Оператор WITH используется для короткого доступа к полям такой переменной.

Тип BITSET

Этот тип данных имеет фиксированный размер N (как правило, N=16 или N=32) и представляет из себя множество целых чисел {0...N-1}, которое описывает множество битов машинного слова компьютера (16 или 32 бита).

Процедура

Параметр-переменная (с идентификатором VAR) не может быть выражением или константой. В качестве результата процедуры-функции не могут выступать: запись, массив, поддиапазон, перечисления, множество. Иначе надо использовать процедуру с параметром-переменной (с идентификатором VAR). Поскольку указатель имеет целочисленный тип, он может быть параметром процедуры и результатом процедуры-функции. Идентификатор VAR говорит о том, что этот параметр может модифицироваться в процедуре и хранить полученный результат (потому что параметр с идентификатором VAR в процедуре представляет собой указатель на данную переменную). Параметр-значение (без идентификатора) представляет собой копию данной переменной, поэтому его изменение в процедуре не приводит к изменению этой переменной.

Экспорт типа записи

Если в модуле определений экспортирующего модуля описан тип указатель на запись, то в импортирующем модуле можно создавать переменные импортированного типа записи без доступа к полям типа записи. Для полного доступа к полям импортированного типа записи в модуле определений экспортирующего модуля должен быть полностью описан тип записи.

Экспорт

В модуле определений должен быть полностью описан экспортируемый тип и/или переменная. При этом условии обеспечивается корректная работа с экспортируемыми переменными и типами в импортирующем модуле. Могут экспортироваться любые типы – записи, массивы, элементарные типы, перечисления, диапазоны и множества.

Оберон, Оберон-2

Тип запись

Может быть пустым (не содержащим полей), поскольку его можно расширить. Базовому типу записи можно присвоить значение расширенного типа записи, но не наоборот, и, аналогично, указателю базового типа записи можно присвоить значение указателя расширенного типа записи, но не наоборот. Это вызвано тем, что при присваивании всем полям переменной типа записи должны быть присвоены значения.

Процедура

Идентификатор VAR говорит о том, что этот параметр может модифицироваться в процедуре и хранить полученный результат (потому что параметр с идентификатором VAR в процедуре представляет собой указатель на данную переменную). Параметр-значение (без идентификатора) представляет собой копию данной переменной, поэтому его изменение в процедуре не приводит к изменению этой переменной. Если используемая в первой процедуре вторая процедура описана ниже первой процедуры, то будет ошибка компиляции. Для правильной компиляции программы необходимо перед первой процедурой записать заголовок второй процедуры, в котором после слова PROCEDURE поставить символ ^, то есть PROCEDURE^.

Процедура, связанная с типом запись

Объявление такой процедуры содержит параметр-приёмник. Связь с типом записи выражается типом приёмника в заголовке объявления процедуры. Приёмник может быть или типом записи в виде параметра-переменной (с идентификатором VAR), или указателем типа записи в виде параметра-значения (без идентификатора). Если процедура связана с базовым типом записи, то она неявно связана с любым расширением этого типа. Однако другая процедура, которая явно связана с расширением базового типа записи, с тем же са-

мым именем, что и процедура, связанная с базовым типом записи, перекрывает её и рассматривается как её переопределение. Формальные параметры обеих процедур должны совпадать. Если к вызову переопределённой процедуры добавить ^, то вызывается процедура, связанная с базовым типом записи, – так называемый супервызов. Осуществить супервызов можно только из переопределённой процедуры. Это связано с инкапсуляцией данных базового типа записи в расширенном типе записи. Недостатком является то, что программист может перепутать при описании процедуры тип параметра-приёмника.

Преимуществами процедур, связанных с типом записи, является то, что потребляемая ими память будет меньше, чем при вызове процедурного типа.

Процедурный тип в типе запись

При его описании в типе записи определяется число и описывается тип параметров. Обязательно в списке параметров фигурирует параметр-переменная, соответствующая указателю на базовый тип записи (без идентификатора). Для каждого расширенного типа записи процедурному типу может назначаться своя процедура (переопределение), в которой число и тип параметров должны совпадать с описанием процедурного типа в базовом типе записи. Поскольку в параметрах процедуры указан базовый тип записи, в теле процедуры обязательно должна быть проверка типа на соответствие данному расширенному типу записи. Преимуществом процедурного типа является отсутствие переопределения процедуры в смысле процедуры, связанной с типом записи, и отсутствие необходимости менять тип параметра-приёмника в описании процедуры, а также нет необходимости в специальных символах для супервызова. Недостатком является возможность забыть присвоить процедуру процедурному типу при создании переменной типа записи.

Проверка типа

Она осуществляется с помощью одного из двух операторов - IS или WITH. Проверка может осуществляться либо сравнением разыменованного указателя с типом записи (базовым или расширенным), что условно может называться проверкой по типу, либо сравнением указателя с указателем на тип записи (базовый или расширенный), что условно может называться проверкой по указателю. Проверка по типу должна осуществляться в отдельной процедуре, поскольку разыменованный указатель или переменная типа записи считаются статическим типом и сравнение с ними надо проводить с передачей параметра-переменной с идентификатором VAR, который преобразует входной статический тип в динамический (который может изменяться), или ещё с идентификатором IN для Компонентного Паскаля. Проверка по указателю может проводиться и без отдельной процедуры, поскольку указатель считается динамическим типом (ему может быть присвоено любое допустимое значение). Оператор IS рассматривает разыменованный указатель p^ как динамический объект, который по адресу в указателе p на момент проверки имеет тип T, и обращается к нему как p^(T), а к полям записи как p^(T).имя поля. Оператор WITH проводит проверку аналогично оператору IS, отличие только в том, что он рассматривает разыменованный указатель p^ как статический объект типа T по адресу в указателе p, и обращается к нему просто как p^, а к полям записи как p^.имя поля, причём запись p^(T) также является допустимой.

Более важным отличием между операторами IS и WITH является то, что оператор IS возвращает логическое значение, поэтому может использоваться в сложных конструкциях проверки условий. Оператор WITH не возвращает логическое значение и не может использоваться в сложных конструкциях проверки условий.

При проверке операторами IS и WITH сравнение надо начинать с проверки на соответствие расширенному типу записи, а в конце проверять на соответствие базовому типу записи, поскольку, если начинать сравнение с проверки на соответствие базовому типу записи, то все расширенные типы базового типа записи будут удовлетворять этой проверке и сравнение будет ограничиваться только этой веткой сравнения, а другие ветки сравнения не будут использоваться.

Экспорт переменных и типов

Экспортируемые переменная и тип становятся глобальными в группе, куда входят модули, которые их импортируют, и модуль, который их экспортирует.

В импортирующем модуле для доступа к полям импортированной переменной типа записи в экспортирующем модуле достаточно экспортировать только поля типа записи. Если в импортирующем модуле необходимо создавать переменные импортированного типа записи, то в экспортирующем модуле необходимо экспортировать тип записи. При этом если не экспортировать поля экспортируемого типа записи, то в импортирующем модуле можно создавать переменную импортированного типа записи, но без доступа к полям типа записи.

Экспорт «только для чтения» используется только для переменных и полей типа записи.

Список экспорта формируется по тексту модуля с обработкой символов, обозначающих экспорт данного типа данных и/или переменной: «*» – экспорт для доступа «чтение/запись», «-» – экспорт для доступа «толь-

ко для чтения». В случае перечислений экспорт переменной и/или этого типа данных влечёт за собой также экспорт всех значений перечислений, что противоречит выбранному типу формирования списка экспорта. В случае диапазонов проблем с экспортом нет, если в качестве базового типа диапазона выступают элементарные типы данных – целые, логические и литерные. В случае, когда базовым типом диапазона выступают перечисления, необходимо экспортировать все значения перечислений, что противоречит выбранному типу формирования списка экспорта. В случае множества проблемы с экспортом те же, что и в случае диапазонов. В Модуле-2 есть предопределённый элементарный тип данных BITSET, имеющий фиксированный размер и использующий в качестве базового типа целые числа. Поэтому этот тип данных используется в Оберон и Оберон-2 как тип данных SET.

В случае, когда в типе запись процедурный тип экспортируется «только для чтения», в импортирующем модуле при расширении данного типа записи эту процедуру нельзя переопределить, а для процедур, связанных с типом записи, как и для обычных процедур, экспорт «только для чтения» запрещён.

Особенности расширения типа записи

Тип записи, содержащий в качестве своего поля другой тип записи, не может считаться его расширением (хотя структура его полей идентична аналогичному расширенному типу записи), поскольку в языке выбрана номинальная (описанная в определении типа связь между типами записи), а не структурная типизация (по совпадению структуры записи).

Компонентный Паскаль

Тип запись

В качестве расширяемого типа записи может быть использован указатель на базовый тип записи.

Атрибут ABSTRACT не позволяет непосредственно создавать переменную такого типа записи (чтобы подчеркнуть его абстрактный характер), но позволяет его расширять (и после этого создавать переменную такого типа записи) и экспортировать.

Атрибут EXTENSIBLE позволяет создавать переменную такого типа записи, расширять его и экспортировать.

Атрибут LIMITED позволяет создавать переменную такого типа записи и расширять его только в том модуле, в котором он определён, а также экспортировать.

Отсутствие атрибутов позволяет создавать переменную такого типа записи, экспортировать его, но не позволяет его расширять.

Процедура

Идентификатор IN говорит о том, что этот параметр служит только для ввода данных и не может быть изменён в процедуре (потому что параметр с идентификатором IN в процедуре представляет собой указатель на данную переменную). Идентификатор IN допускается только для параметров типа запись и типа массив. Константы нельзя помечать идентификатором IN. Идентификатор OUT говорит о том, что этот параметр служит только для вывода результата и его значение в начале процедуры не определено. Всем параметрам с этим идентификатором, содержащим указатели и процедурные типы, в начале процедуры присваивается значение NIL. Поэтому в процедурах, где есть параметры типа указателей и процедурных типов с идентификаторами OUT, необходим хотя бы один оператор присваивания для этих переменных, иначе будет ошибка разыменования указателя NIL. По этой причине процедуры, связанные с типом, с идентификатором EMPTY, не могут содержать параметров с идентификатором OUT. Идентификатор VAR говорит о том, что этот параметр может модифицироваться в процедуре и хранить полученный результат (потому что параметр с идентификатором VAR в процедуре представляет собой указатель на данную переменную). Параметр-значение представляет собой копию данной переменной, поэтому его изменение в процедуре не приводит к изменению этой переменной.

Процедура, связанная с типом запись

Процедура, возвращающая указатель на базовый тип записи, может быть переопределена таким образом, чтобы возвращать указатель на расширенный тип записи и быть связанной с расширенным типом записи.

Атрибут NEW используется для вновь создаваемой для данного типа записи процедуры (её имя должно быть уникальным для базового типа записи и его расширений). Используется для всех атрибутов типов записи.

Атрибут ABSTRACT указывает, что процедура является пустой (не содержит код). Такая процедура ис-

пользуется только для типа записи с атрибутом ABSTRACT и при создании переменной типа записи она должна быть определена.

Атрибут EXTENSIBLE указывает, что процедура может быть переопределена при расширении типа записи и используется для всех атрибутов типов записей.

Атрибут EMPTY указывает, что процедура является пустой (не содержит код). В отличие от процедуры с атрибутом ABSTRACT, при создании переменной типа записи такую процедуру не надо переопределять (или переопределять в случае необходимости). Она используется для типов записи с атрибутами ABSTRACT или EXTENSIBLE, и если она не переопределялась, то её вызов не даёт никакого эффекта. Такие процедуры не могут быть процедурами-функциями и не могут иметь параметров с идентификатором OUT (тело процедуры пустое и не содержит оператора RETURN или присваивания для параметров с идентификатором OUT), но могут содержать параметры-переменные с идентификатором VAR.

При экспорте процедуры «только для чтения» её можно переопределять в импортирующем модуле (если в определяющем её модуле она была описана с атрибутами EXTENSIBLE или EMPTY), но непосредственно запускать её (прямым вызовом) можно только в экспортирующем модуле. Поэтому использовать процедуры с экспортом «только для чтения» в импортирующем модуле можно только из других импортированных процедур, которые экспортированы с доступом «чтение/запись». В связи с этим такой тип экспорта для процедур называется «только для реализации».

Описание свойств различных атрибутов типа записи

Атрибут	Создание переменной типа записи	Расширение	Экспорт
Без атрибута	Да	Нет	Да
ABSTRACT	Непосредственно нельзя, только через расширение	Да	Да
EXTENSIBLE	Да	Да	Да
LIMITED	Только в определяющем этот тип записи модуле	Только в определяющем этот тип записи модуле	Да

Особенности расширения различных атрибутов типа записи

Атрибут расширенного типа записи / Атрибут базового типа записи	Без атрибута	ABSTRACT	EXTENSIBLE	LIMITED
ABSTRACT	Да	Да	Да	Да
EXTENSIBLE	Да	Нет	Да	Да
LIMITED	Нет	Нет	Нет	Только в том модуле, где определён данный тип записи

Сравнение Компонентного Паскаля и Оберонов

Главными являются два новшества – введение указателя как базового типа для расширения типа записи и ослабление правил совместимости типов (эквивалентными являются указатели с эквивалентными базовыми типами данных). Последнее нарушает строгие правила присваивания, которые были в Оберонах. Также в Компонентном Паскале были объявлены устаревшими процедурные типы в записях и был сделан упор на использование процедур, связанных с типом записи. Введённые атрибуты были направлены на то, чтобы сделать их функционал таким же, как для процедурных типов в записи в Оберонах. Рассмотрим более подробно атрибуты типов записей в Компонентном Паскале и сравним их с типами записей в Оберонах. Перед этим надо сделать следующее примечание – все атрибуты типов записей имеют смысл только при их экспорте, а внутри описывающего их модуля они не важны.

Тип записи без атрибута (без возможности его расширения) соответствует экспорту указателя на тип записи с экспортом всех полей этого типа записи (можно создавать экземпляр типа записи, обращаться к её полям, но нельзя его расширять) или переменной этого типа записи.

Тип записи с атрибутом EXTENSIBLE соответствует экспорту типа записи с экспортом всех его полей (можно создавать экземпляр этого типа записи и расширять его).

Тип записи с атрибутом ABSTRACT не имеет смысла без процедур с атрибутом ABSTRACT.

Тип записи с атрибутом LIMITED не имеет смысла экспортировать как указатель, поскольку в импортирующем его модуле нельзя создать экземпляр данного типа записи, но его можно экспортировать как переменную данного типа записи, что также можно сделать и в Оберонах.

Теперь рассмотрим атрибуты процедур, связанных с типом записи, в сравнении с процедурными типами в записях в Оберонах. Здесь также действительно замечание, что эти атрибуты важны только при экспорте содержащего их типа записи.

Процедура с атрибутом NEW соответствует процедурному типу с экспортом с доступом «только для чтения» (её нельзя переопределить при расширении типа записи).

Процедура с атрибутом EXTENSIBLE соответствует процедурному типу с экспортом с доступом «чтение/запись» (её можно переопределить при расширении типа записи).

Процедура с атрибутом EMPTY указывает на пустое тело этой процедуры. Но пустую процедуру можно создать и с атрибутом NEW, при этом компилятор не будет проверять ограничения на типы формальных параметров, как для процедуры с атрибутом EMPTY. Поэтому этот атрибут мог бы использоваться, если бы компилятор проверял процедуры с атрибутом NEW на предмет отсутствия операторов в теле процедуры (она должна содержать хотя бы один оператор).

Процедура с атрибутом ABSTRACT соответствует процедурному типу, которому присвоено значение NIL при создании данного типа записи. При соответствующей доработке компилятора Оберонов можно было бы отслеживать обращение к данному процедурному типу, если его значение равно NIL, и выдавать ошибку на этапе компиляции, как в Компонентном Паскале, а не на этапе работы программы, как это на настоящий момент существует в Оберонах.

Отсюда можно сделать вывод – с точки зрения использования типа запись все соответствующие атрибуты для типов записи и процедур, связанных с типом, сделаны только для того, чтобы можно было использовать тот функционал, что есть в Оберонах для процедурных типов в типах записи.

Использование указателя на тип запись в качестве базового типа для расширения сломало строгую логику экспорта, которая была в Оберонах – если надо было расширить тип запись в импортирующем модуле, то экспортировался тип записи, а если нужно было создать экземпляр типа записи без расширения, то экспортировался или указатель на тип записи, или созданная в экспортирующем модуле переменная типа записи. И это послужило одной из причин введения атрибутов для типов записи.

Единственным положительным отличием Компонентного Паскаля от Оберонов является введение неявного типа String с поддержкой Unicode и более современных диапазонов для числовых типов данных.

Литература

1. Вирт Н. – Программирование на языке Модула-2. Москва, «Мир», 1987.
2. Кристиан К. – Руководство по программированию на языке Модула-2. Москва, «Мир», 1989.
3. Непли Э., Платт Р. – Программирование на языке Модула-2. Москва, «Радио и связь», 1989.
4. Reiser M., Wirth N. – Programming in Oberon. Steps beyond Pascal and Modula. New York, ACM Press, 1992.
5. Мёссенбёк Х., Вирт Н. – Язык программирования Оберон-2. Перевод с английского С. З. Свердлова, Институт компьютерных систем, ETH, Цюрих, 1996.
6. Oberon microsystems, Inc. – Сообщение о языке Компонентный Паскаль. Перевод с английского Ф. В. Ткачёва, 2001.