Программные системы и вычислительные методы
Правильная ссылка на статью:
Дагаев Д.В. Ареальные типы данных в инструментальном подходе к программированию // Программные системы и вычислительные методы. 2025. № 2. DOI: 10.7256/2454-0714.2025.2.73776 EDN: FLOMPB URL: https://nbpublish.com/library_read_article.php?id=73776

Ареальные типы данных в инструментальном подходе к программированию

Дагаев Дмитрий Викторович

ORCID: 0000-0003-0343-3912

Генеральный директор, ООО "СКАДИ"

115230, Россия, г. Москва, Зеленый проспект, 5/12, строение 3, пом. 1

dvdagaev@oberon.org

Статья из рубрики "Языки программирования"

DOI:

10.7256/2454-0714.2025.2.73776

EDN:

FLOMPB

Дата направления статьи в редакцию:

21-03-2025

Аннотация: Для ряда задач классическим методам структурного программирования отдается предпочтение перед методами объектно-ориентированного программирования. Эти предпочтения характерны для детерминированного мира и в системах, ориентированных на машинное представление. Для таких задач разрабатывался модульный язык программирования Оберон, имеющий минималистскую реализацию, существенно отличающуюся от большинства программных средств, стремящихся к максимизации числа поддерживаемых функций. Использование инструментального подхода вместо объектно-ориентированного предлагалось ранее для решения проблем детерминированного мира. Принцип сепарации кода от данных предполагает, что жизненный цикл данных является независимо управляемым, а время жизни превышает по длительности таковое для кода. Данная статья посвящена разработке и анализу ареальных типов данных в контексте инструментального программирования. Ареальные типы данных обеспечивают ортогональную персистентность и интегрируются с кодами, определенными в иерархии типов для инструментов. В качестве метода исследования интеграции ареальных типов используется разработка через тестирование,

реализованная автором в компиляторе МультиОберон. Методы тестирования включали в себя создание тестовых случаев для проверки метаданных ссылок при установлении ареалов, сохранении и восстановлении данных при рестартах ПО. Ареальные ссылки реализованы как персистентные ссылки в ареальном массиве. Благодаря такой организации данных решена проблема сохранения ссылок в структурах данных при завершении и восстановлении ПО. Приведены структуры данных со ссылками, остающимися консистентными при рестартах ПО. Новизна ареальных типов данных заключается в том, что используются механизмы установки ареалов в типах вместо создания объектно-ориентированных разветвленных структур или шаблонов обобщенных классов. Работа со ссылками сочетает преимущества индексов и указателей. Такой подход позволяет реализовывать алгоритмы обобщенного программирования без использования зависимостей по данным, как в части наследования, так в части шаблонов. Приведен пример обобщенного алгоритма сортировки для ареальных типов. Новый тип данных отличается компактностью и структурной простотой по сравнению с динамическими объектами. Преимуществами предложенного подхода являются персистентность, эргодичность, компактность, строгая типизация. В отличие от традиционных подходов объектно-ориентированного и обобщенного программирования ареальные типы данных являются персистентными, не требуют восстановления ссылок после рестарта и используют меньше ресурсов.

Ключевые слова:

ареальные типы, инструментальный подход, Оберон, семантические ограничения, ортогональная персистентность, компилятор, модульность, программирование управляемое данными, информатика-21, метаданные

Введение

С точки зрения теории объектно-ориентированного программирования ООП [1] все пространство задач представляет собой области, где объектная парадигма и парадигма классического структурного программирования разделяются по осям «знания-социальный мир» и «реализация-детерминированный мир» [2]. Для задач детерминированного мира характерен классический подход на основе алгоритмов и структур данных [3]. Автором предложен и используется инструментальный подход к программированию [4], который использует средства классического подхода, но вместо объектов ООП предлагает работу с инструментами. В отличие от объектов, инструменты не содержат в себе данных. Однако, инструменты могут быть ассоциированы с данными. Также, как и объекты, инструменты доступны по указателю, через который возможно реализовать наследование и позднее связывание. В отличие от объектов, каждый инструмент имеет фиксированное расположение в памяти, аллокирование инструментов невозможно, более того, для целей надежности аллокирование запрещается системой семантических ограничений языка МультиОберон [5].

Инструментальный подход построения систем требует правильной организации данных. Он хорошо согласуется с дата-центричным программированием, когда программы отделены от данных и связи между каждым экземпляром структуры данных и программой могут мгновенно и однозначно устанавливаться [6]. Инструментальный подход содержит коды программ, но не содержит данных. Целью представленной работы является определение структур данных, требуемых для инструментального подхода.

Для реализации требований детерминированных систем 24x7 необходимо решение ряда задач в части структур данных. Во-первых, обеспечение персистентности, где время жизни данных может существенно превышать время жизни программ. Во-вторых, нужен ссылочный доступ к данным различных типов. В третьих, необходим эффективный доступ к данным из инструментов.

Предметом исследования статьи являются предлагаемые ниже ареальные структуры данных. В качестве метода исследования используется разработка через тестирование, реализованная автором в компиляторе МультиОберон.

Персистентность данных и ортогональная персистентность

В объектных системах возникает проблема обеспечения хранения и восстановления состояния. Она решается методом сериализации при сохранении и десереализации при восстановлении. Разработчики стоят перед необходимостью обеспечить корректную сериализацию и десериализацию даже при сбоях в работе ПО.

В дата-центричных системах данные отделены от кода, время жизни данных существенно больше времени жизни кода. Данные сохраняются и восстанавливаются независимо от работы кода. Системы, отвечающие принципу «ортогональной персистентности» [7], имеют механизмы независимого сохранения состояния данных, например, как для Java-системы в [8] или в реализации на языке Motoko [9].

Задачей персистентности является максимально приблизить формат хранения данных к формату представления в памяти. Задачей ортогональности является реализация независимого от кода механизма преобразования из данных в файловой системе в данные в памяти.

Организация ссылок в данных требует использования указателей на динамически выделяемые структуры. Указатели являются временными адресами в оперативной памяти, не подлежащими сериализации. Восстановление ссылок требует очередного аллокирования и подмены адресов во всех точках со ссылками.

Для данных, представленных в виде массивов, возможно использование целочисленных индексов, но работа с индексами требует явного указания массива, к которому индекс относится.

Ареальные структуры данных

В данной статье предлагаются ареальные структуры данных – запись и ареальная ссылка. Ареальная запись означает тип данных запись, которая не может иметь указателей, но может иметь ссылки на ареальные объекты. Ареал означает определенное место обитания и представляет собой массив из ареальных объектов. Ареальная ссылка содержит не адрес, а индекс объекта в ареале, но, поскольку она имеет определенный тип, то ареальная ссылка путем тривиального преобразования разрешается как адрес ареальной записи.

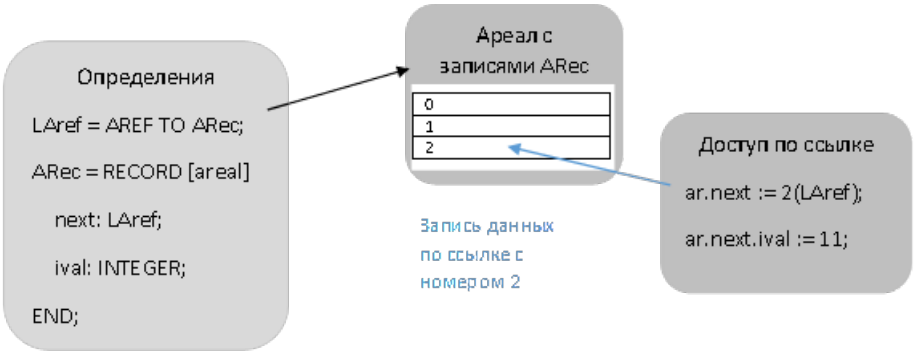

Рис. 1. Ареальная запись и ареальная ссылка

Fig.1 Areal record and areal reference

Ареальные структуры данных в паскале-подобной форме даются в нотации языка Оберон [10] и реализованы в системе МультиОберон. Запись ареальная отличается от стандартной записи наличием флага [areal]. Ареальная ссылка является ссылкой на объект ареальной записи.

LAref = AREF TO ARec;

ARec = RECORD [areal]

next: LAref;

ival: INTEGER;

END;

Тип ареальной ссылки аналогичен типу адреса структуры. Он представляет собой целочисленный тип длины адреса, привязанный к типу адресуемой структуры. Но значение ареального типа отличается. Если для указателя значения адреса инкрементируются размером указателя (4 для 32-битных систем и 8 для 64-битных), то для ареального типа значения ареальной ссылки могут принимать величины 0, 1, 2, 3, N, где N-размер выделенного массива под ареал.

Учитывая вышесказанное, установка значения ссылок аналогична установке значений целочисленных величин. Только необходимо правильное преобразование типов. Например, установка ссылке значения 2 будет выглядеть следующим образом.

ar.next := 2(LAref);

Нулевая ссылка представляет собой 0 и преобразуется в адрес элемента ареала с индексом 0. Поэтому чтение-запись по нулевой ареальной ссылке допустимы. Разумеется, нулевую ссылку можно использовать в алгоритмах специальным образом, как это применяется в указателях.

Доступ по ссылке синтаксически подобен доступу по указателям. В нашем примере можно присвоить элементу данных значение 11.

ar.next.ival := 11;

Семантически такой доступ означает присвоение 11 элементу массива ареала.

__areal_internal[(*index*)2].ival := 11;

По типу ссылки находится массив ареала __areal_internal, а по значению ссылки находится индекс элемента 2.

Установка массива ареала

Для ареального типа может быть установлен один и только один массив. Должна быть представлена информация о типе данных, адресе начала массива и числе элементов в нем. Встроенная функция AREAL устанавливает массив, связанный с ареальным типом.

AREAL(ARec, array[(*index*)0], LEN(array));

Значения адреса начала и длины массива записываются в метаданные указанного типа ARec.

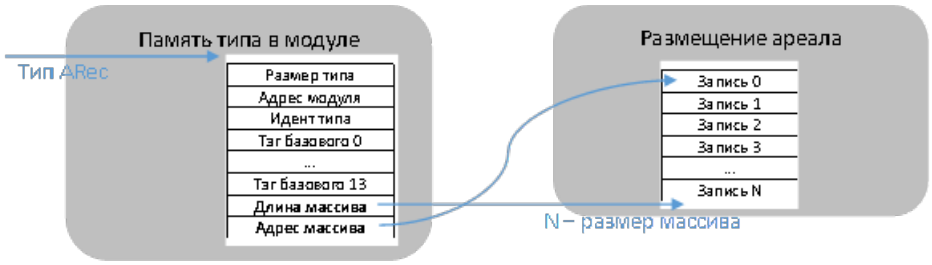

Рис. 2. Метаданные ареальной записи

Fig.2 Areal record metadata

В метаданных записи содержится массив указателей на базовые типы, элемент 14 используется для сохранения длины, а элемент 15 используется для сохранения адреса массива ареала.

Значение длины массива доступно через встроенную функцию LEN. Получение длины ареальной ссылки записывается следующим образом.

VAR ar: LAref;

l := LEN(ar);

Функция получения длины ареала зависит только от типа данных. Поэтому длина возвращается корректно для нулевой, для инициализированной и даже для неинициализированной ссылки.

Персистентность данных ареала

Данные ареала могут сохраняться без преобразования и без преобразования восстанавливаться. До сохранения ареальный массив находится в одном пространстве адресов процесса 1. А после восстановления ареальный массив находится в другом пространстве адресов процесса 2.

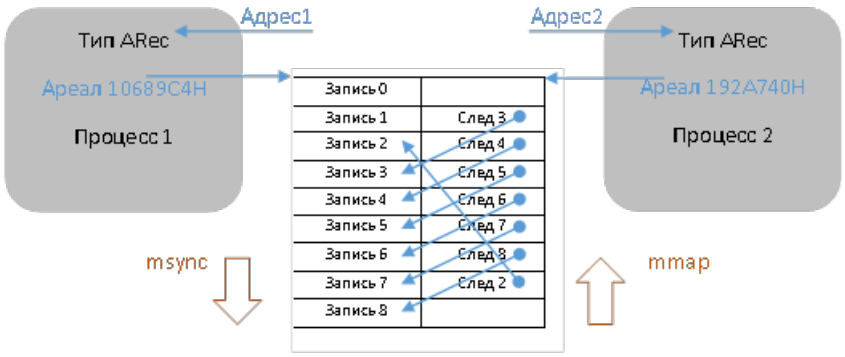

Рис. 3. Ареал в разных пространствах адресов

Fig.3 Areal in different address spaces

Сохранение/восстановление в файл без преобразования в POSIX-системах осуществляется вызовами msync/mmap. Если файл предоставляется в общий доступ через разделяемую память, то процессы могут работать с данными файла одновременно. При очередной загрузке данных из файла необходимо устанавливать массив ареала. В нашем примере для первого процесса и для второго процесса это будут следующие выражения.

AREAL(ARec, 10689C4H, 9);

AREAL(ARec, 192A740H, 9);

Таким образом, ареалы типов могут быть поставлены в разделяемый разными процессами доступ. При этом структуры метаданных с типом ARec двух процессов расположены отдельно и загружаются каждый в свое адресное пространство (на рисунке 3 Адрес1 для 1 процесса и Адрес2 для второго). Изменения в ареале будут доступны всем процессам, изменения метаданных (например, уменьшение длины ареала) – будут доступны только текущему процессу.

Структура файлов данных ареала

Для загрузки массивов значений нескольких ареалов из файла используется следующий цифровой формат данных.

Рис. 4. Формат файла ареала

Fig.4 Areal file format

Каждая запись о массиве ареала предваряется длиной массива и тэгом типа Оберона [11]. Тэг типа представляет собой контрольную сумму из имени типа, имен и типов всех полей в порядке вхождения. Формируется во время компиляции. В структуре загруженного типа обязательно присутствует тэг. Если при загрузке данных обнаружено

несовпадение данных с тэгом типа, то данные считаются неконсистентными и генерится сообщение об ошибке. Длина ареала устанавливается <длина ареала> := <длина> / <размер типа>.

После корректной загрузки данных все ареальные массивы получают тип и длину. Если массивы не инициализированы, длина ареала будет нулевой.

Трансформация отношений обобщения

Отношение обобщения в UML описывает иерархическое строение классов и наследование их свойств и поведения [12]. В диаграмме классов класс-потомок обладает всеми свойствами и поведением класса-предка, а также имеет свои собственные свойства и поведение, которые могут отсутствовать у класса-предка.

Примером реализации отношений может служить сортировка, алгоритм которой относится к общему элементу, а реализации сравнений и перестановки – к частному. При этом общие алгоритмы расширяются в частных реализациях.

Для случая ООП реализация основывается на включении алгоритмов сортировки в базовый класс, а сравнений и перестановок – в специфический[13]. По сравнению с ООП инструментальный подход использует как ареальные записи, так и базовые и расширенные инструменты.

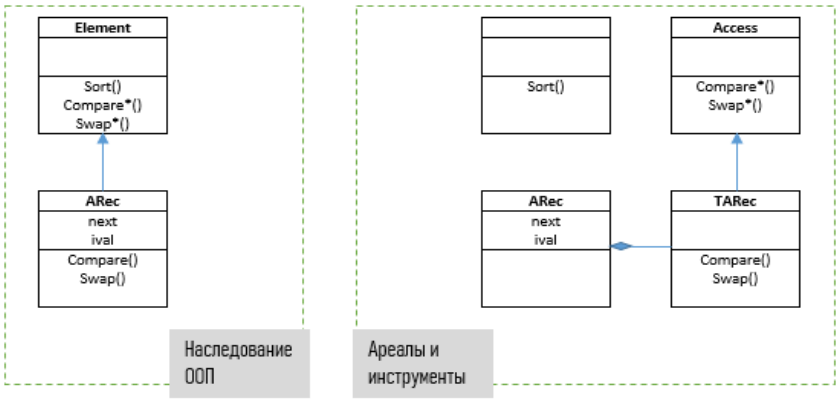

Рис. 5. Отношения обобщения

Fig.5 Generalization relationships

Базовый инструмент Access содержит определения методов доступа к сортируемым структурам данных. На уровне обобщения нам известно, что инструмент имеет две операции сравнения и перестановки.

Access = POINTER TO ABSTRACT RECORD [tool] END;

PROCEDURE (acc: Access) Compare (l1, l2: INTEGER): INTEGER, NEW, ABSTRACT;

PROCEDURE (acc: Access) Swap (l1, l2: INTEGER), NEW, ABSTRACT;

Обобщенный алгоритм сортировки требует наличия экземпляра базового инструмента acc, длины A_LEN и не требует никакой информации о типе данных реализации.

TOOL(acc);

FOR j := 1 TO A_LEN-1 DO

```
FOR k := A_LEN-1 TO j BY -1 DO

IF acc.Compare(k-1, k) < 0 THEN

acc.Swap(k-1, k)

END

END

END;
```

Реализация массива основывается на ареальном типе данных ARec, описанном выше. Тип и длина должны быть установлены.

```
VAR arrARec = ARRAY A_LEN OF ARec;

AREAL(ARec, arrARec[(*index*)0], A_LEN);
```

Реализация методов сравнения и перестановки в специфическом ARecAccess инструменте привязана только к ареальным данным типа ARec.

```
ARecAccess = POINTER TO RECORD [tool] (Access) END;

PROCEDURE (acc: ARecAccess) Compare (l1, l2: INTEGER): INTEGER;

VAR lar1, lar2: LARef;

BEGIN

lar1 := l1(LARef);

lar2 := l2(LARef);

RETURN lar1.ival-lar2.ival;

END Compare;

PROCEDURE (acc: ARecAccess) Swap (l1, l2: INTEGER);

VAR lar1, lar2: LARef; x: ARec;

BEGIN

lar1 := l1(LARef);

lar2 := l2(LARef);

x := lar1^;

lar1^ := lar2^;

lar2^ := x;

END Swap;
```

Следует отметить, что обобщенные алгоритмы приведенного вида не накладывают никаких ограничений на данные. Единственным исключением является то, что данные должны быть ареального типа. Это объясняется тем, что инструменты не содержать указателей на данные, и единственно, что можно извлечь из специфического

инструмента – это тип. И только для ареального типа можно получить массив значений и длины, с этим типом ассоциированные. Такой подход содержит меньше внутренних зависимостей, чем ООП[14-15], накладывающие ограничения на необходимость наличия базового класса в сортируемых структурах данных.

Реализация в обобщенном программировании

Подход обобщенного программирования в классическом [16] и современном [17] вариантах приводит к разработке шаблонов обобщенных алгоритмов и реализации специфических алгоритмов для каждого типа данных. Аналогичный пример содержал бы шаблон алгоритма аналогичный приведенному.

```
template

E bsort(E a, int len) {

for (int j = 1; j < len; j++) {

for (int k = len-1; k >= j; k--)

if (a[k-1] < a[k])

std::swap(a[k-1], a[k]);

}
```

Подход обобщенного программирования также требует необходимости определения операций сравнения и копирования для используемых типов. Синтаксис и семантика программирования с шаблонами существенно усложняет использование. Обобщенное программирование является избыточным, т.к., помимо обобщенного алгоритма в выходной программе содержится тот же алгоритм, реализованный для специфического типа.

Для более сложных структур данных (отображения, деревья, списки) работа в стиле обобщенного программирования требует использования итераторов. Такие структуры организуются в динамической памяти, и для эффективной навигации итераторы становятся необходимыми. Структуры обобщенного программирования не являются персистентными, для них необходимы сложные рекурсивные алгоритмы сериализации/десереализации.

Ареальные структуры данных используют тривиальное преобразование индекс – адрес для всех используемых структур.

Доступ к полям ареальных записей

Ареальная ссылка, имеющая доступ к внутренним полям ареального типа, должна быть определена в том же модуле, что и тип. Попытка сослаться на тип из другого модуля приводит к ошибке компиляции. При этом ареальная ссылка может не экспортироваться, даже если экспортируется сам тип. Поэтому можно возвращать значения элементов ареала без доступа к самому ареалу. Например, возвращаемый функцией GetVal элемент Value никаких ссылок на ареал не содержит.

```
AVal = AREF TO Value;

Value* = RECORD [areal]
```

```
rval-: REAL;

ival-: INTEGER;

END;

PROCEDURE GetVal*(select: INTEGER; VAR v: Value);
```

Приведенный подход имеет недостаток в том, что информацию об ареале нельзя извлечь из значения, но можно извлечь из типа Value. Поэтому при необходимости нужно разделять ареальный и возвращаемый типы.

Ареальные данные могут содержать как информацию, необходимую для пользователя, так и специфическую информацию, привязанную к реализации. В этом случае необходимо разбивать данные на базовые и производные типы, структурно отделяя передаваемую информацию от скрытой, рисунок 6.

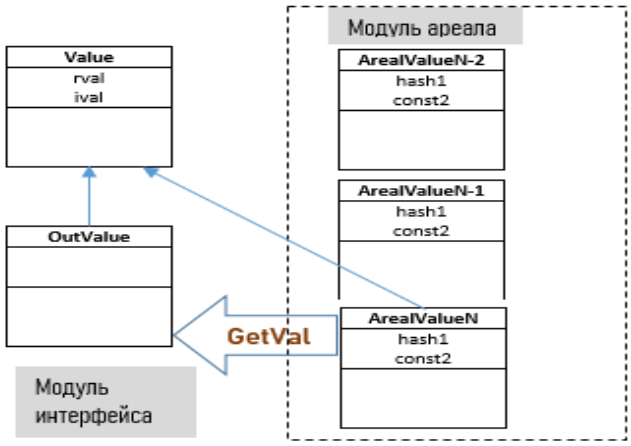

Рис. 6. Разделение типов и модулей

Fig.6 Modules and types separation

В приведенном примере Value является базовым типом, от которого наследуются пользовательский OutValue и скрытый ArealValue. Последний имеет дополнительные поля hash1, const2.

```
(* interface module *)

Value* = EXTENSIBLE RECORD

rval*: REAL;

ival*: INTEGER;

END;

OutValue* = RECORD (Value)

END;

(* areal module *)

AVal = AREF TO ArealValue;

ArealValue = RECORD [areal] (I.Value)
```

```
hash1*: INTEGER;

const2*: REAL;

END;

PROCEDURE GetVal*(select: INTEGER; VAR v: I.Value);
```

При запросе данные из ArealValue, относящиеся к базовому типу Value, копируются функцией GetVal в подаваемую на вход структуру OutValue. В данном случае возвращаемый тип OutValue не имеет информации об ареальном типе ArealValue.

Семантические ограничения ареалов

Поскольку использование ареалов и инструментов является расширением языка Оберон, то включение механизмов с флагами [area], [tool] требует снятия семантического ограничения на инструменты. Это делается следующим оператором.

```
RESTRICT +AREAL, +TOOL
```

Следствием применения ареалов является запрет выделения и использования динамической памяти. В системе МультиОберон указанные запреты означают, что в модуле непосредственно функция NEW быть вызвана не может. Кроме этого, не используются указатели на выделенную память, а используются ареальные ссылки. Также запрещается использование системных преобразований и доступ к метаданным.

```
RESTRICT –NEW, -POINTER, -SYSTEM;
```

При наличии ключевых слов NEW, POINTER и SYSTEM проверка не будет пройдена и завершится синтаксической ошибкой.

Вопросы эргодичности

Деградации данных со временем, определяющей свойства эргодичности [18], в ареальных записях не происходит. Однако необходимо обеспечить неизменность набора данных в течение жизненного цикла системы.

В системах детерминированного мира выделяется избыточное количество данных: помимо необходимых на данный момент, выделяются резервные данные. При добавлении нового элемента в реальности происходит выбор резервного элемента.

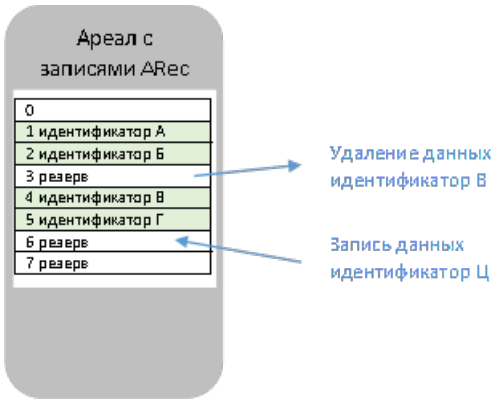

Рис. 7. Добавление резервного элемента

Fig.7 Adding reserve element

Данные ареальных типов со временем не появляются, не исчезают, а только меняют свое состояние. Состояние, определяющее является ли указанные элемент резервным, являются важным и персистентным параметром[19]. Система адресации записей не меняется по ходу работы. Следовательно, необходимо только обосновывать базовые алгоритмы в части правильности значений данных в ареальных типах.

Разделение кода и данных означает, что оценку вклада кода можно делать только в том случае, когда идет обновление данных.

Приведем доказательства эргодичности построенной на указанных принципах системы. В части управления памятью следует отметить, что динамические структуры данных не используются, а память ареальных массивов сохраняется на протяжении времени жизни данных. В части обеспечения реального времени инструментальный подход не вносит дополнительных конструкций. В части обработки исключительных ситуаций происходит завершение программного кода. При этом не требуется сериализация и десереализация при восстановлении данных. Ареальные записи могут передаваться как сообщения при сетевом обмене данных. При этом ареальные ссылки будут сохраняться. Инструментальный подход не влияет на управление временем.

Сравнение подходов

Ареальные типы инструментального подхода реализуют персистентность ссылок и сепарацию кода от данных. Решение реальных задач детерминированного приводит к построению связанных структур данных в компактном и возобновляемом состоянии. Современные объектные облачные системы из «социального мира» выстраивают достаточно сложные механизмы агрегирования объектов, их взаимодействия, децентрализации и механизмов диагностики и восстановления, распределения нагрузки и т.д.[20]. При этом решение этими методами задач детерминированного мира приводит к огромной избыточности, потерям надежности во всех режимах, отсутствию эргодичности.

В части алгоритмов работы с данными инструментальный подход не накладывает на данные каких-либо ограничений (наследование от одного и того же типа, необходимость итераторов). Требования для выполнения операций с данными относятся к кодам инструментов. В части реализации стандартные алгоритмы библиотек шаблонов в большей степени интегрированы с оптимизирующими компиляторами.

Ареальные типы имеет смысл использовать в системах, где реализуется отказ от динамических структур данных и динамического выделения памяти в целом. Стандартные динамические типы данных обладают большей гибкостью и возможностью адаптации под изменяющиеся структуры предметной области.

Применение в ПО моделирующих систем

Ареальные типы данных использовались при разработке моделирующего комплекса ПО СКОЯТ, расшифровываемого как система комплексного обучения ядерным технологиям. Гибридная архитектура СКОЯТ состоит из облачных подсистем управления ресурсами (включая сервисы, контейнеры, распределение нагрузок) и подсистем СКОЯТ-МОДЕЛЬ, организованных на принципах детерминированного мира.

Каждая подсистема из набора моделей, запускаемая внутри контейнера, включат в себя совокупность взаимосвязанных ареальных структур данных.

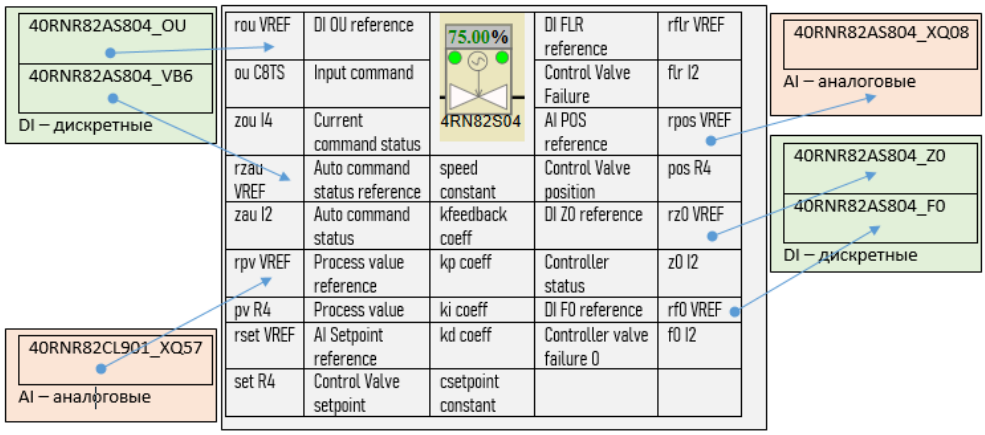

Рис. 8. Привязка данных регулятора СКОЯТ

Fig.8 SCONT controller data separation

Каждый регулятор представляет собой элемент структуры данных, имеющий идентификацию и привязки к другим элементам. Каждый элемент входных и выходных данных представляет собой поле ареальной записи со ссылкой на элемент таблицы аналоговых или дискретных параметров. Ссылки хранятся персистентно в том же виде, что и в памяти. Параметры, относящиеся к данному экземпляру оборудования, также являются персистентными.

Программные коды расположены в динамически загружаемом плагине обработки для объекта регулятора. Аналогично устроены коды и данные плагинов обработки объектов иных типов (задвижки, насосы, блокировки, и т.д.).

Заключение

Предложенные в данной статье ареальные типы данных позволяют организовать механизм ортогональной персистентности данных по отношению к кодам программ. Ареальные типы данных предназначены и используются в инструментальном подходе к программированию. Ареальные типы и данных и инструментальной подход реализуют инфраструктуры программирования управляемого данными. В рекомендуемом для систем детерминированного мира подходе используется сепарация кода от данных и персистентность данных.

Сохранение и восстановление ареальных данных не требует операций сериализации/десериализации. Ареальные данные хранятся в том виде, в котором они используются в памяти. Коды программ инструментального подхода не хранят внутри себя состояний. Следовательно, нет необходимости сохранять и восстанавливать что-нибудь в кодах, если данные используются персистентно.

Ареальные типы обеспечивают ссылочный типизированный доступ к данным. Массив ареальных данных существует в единственном экземпляре в приложении. Ссылка на элемент массива является ссылкой на ареальную запись. Ссылка содержит целочисленный индекс и однозначно идентифицирует конкретную ареальную запись при любых использованиях.

Принцип сепарации кода от данных предполагает использование инструментов с ареальными типами данных. Инструменты реализуют набор методов процедур без использования понятия экземпляров. Они могут быть привязаны к определенному типу данных.

Недостатком ареального типа данных являются ограниченность области применения, в части сравнения с ООП[21]. Также недостатком является невозможность дублирования ареалов одного и того же типа, правда эта проблема решается на уровне базовых и производных типов, как показано выше.

Ареальные типы данных реализованы в системе МультиОберон. Ареальные типы данных интегрированы с инструментальным подходом. Реализован эффективный доступ из инструментов путем преобразования индекса в ареальную ссылку.

Ортогональная персистентность означает, что все значения полей структур данных обновляются, сохраняются и восстанавливаются в персистентной памяти. Такой подход обеспечивает надежную и безударную работу системы при сбоях и отказах в программных кодах[22]. Такая организация данных отвечает требованиям эргодичности разрабатываемых систем. Программные системы, удовлетворяющие указанным требованиям, не деградируют со временем из-за изменения состояния внутренней памяти.

Применение в моделирующем комплексе системе обучения СКОЯТ реализовано с использованием ареальных структур данных. Решение об использование позволило не только апробировать новый метод, но и обеспечить большую понятность и консистентность структур данных и их связей на протяжении всего жизненного цикла разработки ПО.

Новизна ареальных типов данных заключается в том, что используются механизмы установки ареалов в типах вместо создания объектно-ориентированных разветвленных структур или шаблонов обобщенных классов. Такой подход позволяет сочетать преимущества индексов и указателей.

Значимость предложенного подхода заключается в изменении структуры программного обеспечения, более соответствующего задачам детерминированного мира. Преимуществами построения систем на основе инструментального подхода с ареальными типами являются персистентность, эргодичность, компактность, строгая типизация.

Библиография

1. Mansi Dhirajsinh Parmar, Sarthavi Parmar. Survey on Concept of Object-Oriented Programming. International Journal of Scientific Research in Computer Science Engineering and Information Technology. 2024. Vol. 10, No. 2. P. 427-431. DOI: 10.32628/CSEIT243647. EDN: AMCTWA.
2. West D. Object Thinking. Redmond: Microsoft Press, 2004.
3. Вирт Н. Алгоритмы и структуры данных. М.: ДМК-Пресс, 2016.
4. Дагаев Д.В. Инструментальный подход к программированию в системе МультиОберон // Программные системы и вычислительные методы. 2024. № 1. С. 31-47. DOI: 10.7256/2454-0714.2024.1.69437 EDN: WVZVVU URL: https://nbpublish.com/library_read_article.php?id=69437
5. Дагаев Д.В. Ограничительная семантика языка в системе МультиОберон // Программные системы и вычислительные методы. 2023. № 1. С. 26-41. DOI: 10.7256/2454-0714.2023.1.36217 EDN: IWIODR URL: https://nbpublish.com/library_read_article.php?id=36217
6. Rajive J., Ph.D. Data-Oriented Architecture: A Loosely-Coupled Real-Time SOA. Real-Time Innovations, Inc., 2007.
7. Hosking A., Nystrom N., Cutts Q., Brahnmath K. Optimizing the read and write barriers

for orthogonal persistence // Advances in Persistent Object Systems / Morrison J., Atkinson M. (Eds.). San Francisco: Morgan Kaufmann, 1999. P. 11.
8. Lefort A. A Support for Persistent Memory in Java. Computer science. Institut Polytechnique de Paris, 2023. English. ffNNT : 2023IPPAS001.
9. Bläser L., Russo C., Greif G., Vandersmith R., Ibrahim J. Smarter Contract Upgrades with Orthogonal Persistence // Proceedings of the 16th ACM SIGPLAN International Workshop on Virtual Machines and Intermediate Languages (VMIL '24). Pasadena, CA, USA, 2024. P. 11.
10. Вирт Н., Гуткнехт Ю. Разработка операционной системы и компилятора. Проект Оберон. М.: ДМК-Пресс, 2015.
11. Crelier R. OP2: A Portable Oberon Compiler. ETH, Departement Informatik, 1990. P. 32-34.
12. Леоненков А.В. Самоучитель UML. 2-е изд., перераб. и доп. СПб.: БХВ-Петербург, 2006. P. 150.
13. Гамма Э., Хэлм Р., Джонсон Р., Влиссидес Дж. Приемы объектно-ориентированного проектирования. Паттерны проектирования. СПб.: Питер, 2001.
14. Мартин Р. Чистая архитектура. Искусство разработки программного обеспечения. СПб.: Питер, 2022.
15. Мартин Р. Идеальный программист. Как стать профессионалом разработки ПО. СПб.: Питер, 2022.
16. Musser D.R., Stepanov A.A. Generic Programming // Proceedings of the International Symposium ISSAC'88 on Symbolic and Algebraic Computation. New York: Springer-Verlag, 1988. P. 13-25.
17. Roynard M. Generic programming in modern C++ for Image Processing. Signal and Image Processing. Sorbonne Université, 2022. English. ffNNT : 2022SORUS287ff.
18. Дагаев Д.В. О разработке Оберон-системы с заданными свойствами эргодичности // Труды ИСП РАН. 2020. Т. 32, вып. 6. С. 67-78. DOI: 10.15514/ISPRAS-2020-32(6)-5. EDN: HBGGIK.
19. Дагаев Д.В. Исполняющая машина автоматных программ // Научно-технический вестник информационных технологий, механики и оптики. 2021. Т. 21, № 4. С. 525-534. DOI: 10.17586/2226-1494-2021-21-4-525-534. EDN: MJZXQZ.
20. Lo S.K., Lu Q., Zhu L., Paik H.-Y., Xu X., Wang C. Architectural patterns for the design of federated learning systems // Journal of Systems and Software. 2022. Vol. 191. P. 357.
21. Ali H.M., Hamza M.Y., Rashid T.A. Exploring Polymorphism: Flexibility and Code Reusability in Object-Oriented Programming // Proceedings of the 4th International Conference on Recent Innovation in Engineering (ICRIE 2023). University of Duhok, College of Engineering, 13th-14th September 2023. Paper No. 33.
22. Khattak U.F., Hussein H.A. A review on graph representation for object-oriented programming // BIO Web of Conferences. 2024. Vol. 97. Article 00131. DOI: 10.1051/bioconf/20249700131. EDN: YIHEDW.

Результаты процедуры рецензирования статьи

В связи с политикой двойного слепого рецензирования личность рецензента не раскрывается.
Со списком рецензентов издательства можно ознакомиться здесь.

Предметом исследования в рецензируемой работе выступают Ареальные типы данных, рассматриваемые в инструментальном подходе к программированию.
Методология исследования базируется на инструментальном подходе к построению систем и программированию, использовании языка Оберон и системы МультиОберон. Актуальность темы авторы работы связывают с тем, что в системах детерминированного

мира, где время жизни данных может существенно превышать время жизни программ, важно реализовать персистентность данных и эффективный доступ к данным различных типов.

Научная новизна рецензируемого исследования, к сожалению в статье четко не сформулирована авторами.

В тексте статьи выделены следующие разделы: Введение, Персистентность данных и ортогональная персистентность, Ареальные структуры данных, Установка массива ареала, Персистентность данных ареала, Структура файлов данных ареала, Трансформация отношений обобщения, Реализация в обобщенном программировании, Доступ к полям ареальных записей, Семантические ограничения ареалов, Вопросы эргодичности, Сравнение подходов, Применение в ПО моделирующих систем, Заключение и Библиография.

В статье изложено авторское видение ареальных структур данных, при подаче материала широко использован графический способ – текст иллюстрирован 8 рисунками, кроме этого приводятся фрагменты программного кода, проведено сравнение объектно-ориентированного и инструментального подходов в программировании, отмечено, что ареальные типы имеет смысл использовать в системах, где реализуется отказ от динамических структур данных и динамического выделения памяти в целом.

Библиографический список включает 16 источников – научные публикации по рассматриваемой теме в российских и зарубежных журналах на русском и английском языках. В тексте приведены адресные ссылки, что подтверждает наличие апелляции к оппонентам.

Из замечаний стоит отметить следующие. Во-первых, во введении не отражены такие элементы методологического аппарата любого научного исследования как цель и задачи, предмет и объект, цель, описание исходных материалов и методов, с помощью которых они преобразованы в результаты, новизна и значимость полученных результатов. Во-вторых, в тексте употребляются аббревиатуры без их расшифровки, например, ООП – представляется, что среди читателей могут быть и те, кому будет непонятно, что речь идет об объектно-ориентированном программировании. В-третьих, уместно расширить список источников до рекомендуемого издательством количества – не менее 20 источников, не менее половины работ, изданных в последние 3 года. В-четвертых, некоторые фрагменты текста статьи уже были опубликованы автором ранее, например, значительная часть введения повторяет введение статьи Дагаев Д. В. Инструментальный подход к программированию в системе МультиОберон / Программные системы и вычислительные методы. 2024. № 1. DOI: 10.7256/2454-0714.2024.1.69437 EDN: WVZVVU URL: https:/nbpublish.com/library_read_article.php?id=69437.

Рецензируемый материал соответствует направлению журнала «Программные системы и вычислительные методы», отражает результаты проведенной авторами работы, может вызвать интерес у читателей, но нуждается в доработке перед опубликованием.

Результаты процедуры повторного рецензирования статьи

В связи с политикой двойного слепого рецензирования личность рецензента не раскрывается.

Со списком рецензентов издательства можно ознакомиться здесь.

Представленная статья на тему «Ареальные типы данных в инструментальном подходе к программированию» соответствует тематике журнала «Программные системы и вычислительные методы» и посвящена актуальному вопросу – применению инструментального подхода к программированию с использованием средств классического подхода, но вместо объектов объектно-ориентированного

программирования осуществлению работы с инструментами.

В статье четко обозначена цель исследования – определение структур данных, требуемых для инструментального подхода.

В статье авторы указывают в качестве предмета исследования предлагаемые авторские ареальные структуры данных. В качестве метода исследования авторы используют разработку через тестирование, реализованную автором в компиляторе МультиОберон.

Также в статье указана теоретико-методологическая основа исследования, а именно авторы ссылаются на работы Hosking A., Nystrom N., Cutts Q., Brahnmath K. (в работе рассматриваются вопросы системы, отвечающие принципу «ортогональной персистентности»), Lefort A. (рассматривает вопрос механизмов независимого сохранения состояния данных, например, как для Java-системы).

Авторы в данной статье предлагают ареальные структуры данных – запись и ареальная ссылка. В статье результаты исследования авторы представляют в графическом виде.

В списке литературы имеются российские и зарубежные источники по теме исследования. Стиль и язык изложения материала является научным и доступным для широкого круга читателей. Статья по объему соответствует рекомендуемому объему от 12 000 знаков.

Статья достаточно структурирована - в наличии введение, заключение, внутреннее членение основной части (в статье рассмотрены следующие вопросы: персистентность данных и ортогональная персистентность, ареальные структуры данных, установка массива ареала, персистентность данных ареала, структура файлов данных ареала, трансформация отношений обобщения, реализация в обобщенном программировании, доступ к полям ареальных записей, семантические ограничения ареалов, вопросы эргодичности, применение в ПО моделирующих систем).

Практическая значимость статьи четко обоснована. Значимость предложенного авторами подхода заключается в изменении структуры программного обеспечения, более соответствующего задачам детерминированного мира. Преимуществами построения систем на основе инструментального подхода с ареальными типами являются персистентность, эргодичность, компактность, строгая типизация.

В заключении авторы акцентируют внимание на том, что предложенные в данной статье ареальные типы данных позволяют организовать механизм ортогональной персистентности данных по отношению к кодам программ. Ареальные типы данных предназначены и используются в инструментальном подходе к программированию. Ареальные типы и данных и инструментальной подход реализуют инфраструктуры программирования, управляемого данными. В рекомендуемом авторами для систем детерминированного мира подходе используется сепарация кода от данных и персистентность данных.

Статья «Ареальные типы данных в инструментальном подходе к программированию» может быть рекомендована к публикации в журнале «Программные системы и вычислительные методы».