Обобщенное программирование

материал взят из книги Александра Степанов "Начала программирования"

18 марта 2026 г.

Регулярные процедуры

Процедура является **регулярной**, если и только если замена ее входных объектов равными объектами приводит к получению равных выходных объектов.

Регулярные процедуры

Функциональная процедура — это регулярная процедура, определенная на регулярных типах, с одним или несколькими непосредственно передаваемыми входными объектами и единственным выходным объектом, возвращаемым как результат процедуры.

Регулярные процедуры

Если размер параметра является небольшим или если процедуре требуется копия, в которую она могла бы вносить изменения, передаем параметр по значению, выполняя локальную копию. В противном случае параметр передается по неизменяющейся ссылке.

Регулярные процедуры

Функциональная процедура может быть реализована как функция C++, указатель на функцию или функциональный объект.

Функциональная процедура

```
int plus_0(int a, int b)
{
        return a + b;
}
```

Это – семантически эквивалентная функциональная процедура:

```
int plus_1(const int& a, const int& b)
{
        return a + b;
}
```

Регулярные процедуры

Это — семантически эквивалентная, но не функциональная процедура, поскольку ее входные и выходные объекты не передаются непосредственно:

```
void plus_2(int* a, int* b, int* c)
{
        *c = *a + *b;
}
```

В plus2 объекты a и b являются входными, тогда как c — выходной объект. Понятие функциональной процедуры — это синтаксическое, а не семантическое свойство: в нашей терминологии процедура plus2 является регулярной, но не функциональной.

Регулярные процедуры

Областью определения функциональной процедуры является подмножество значений ее входных объектов, для применения к которому она предназначена.

Регулярные процедуры

Однородная функциональная процедура — это такая функциональная процедура, все входные объекты которой имеют одинаковый тип.

Регулярные процедуры

Домен однородной функциональной процедуры представляет собой тип ее входных объектов.

Регулярные процедуры

Кодомен для функциональной процедуры представляет собой тип его выходного объекта.

Регулярные процедуры

Область значений для функциональной процедуры представляет собой множество всех значений из ее кодомена, возвращенное процедурой применительно к входным объектам из ее области определения.

Регулярные процедуры

```
int square(int n){ return n * n; }
```

В то время как ее домен и кодомен имеют тип int, ее областью определения является множество целых чисел, квадрат которых является представимым с помощью этого типа, а областью значений — множество квадратов целых чисел, представимых с помощью того же типа.

Концепции

Повышению полезности программного компонента, такого как библиотечная процедура или структура данных, способствует его разработка не в терминах конкретных типов, а в терминах требований к типам, выраженных как синтаксические и семантические свойства. Мы называем коллекцию требований концепцией. Типы представляют виды, концепции — роды.

Концепция

Чтобы мы могли описывать концепции, требуется несколько механизмов работы с типами: атрибуты типов, функции и конструкторы типа.

Концепция

Атрибут типа — это отображение из типа в значение, описывающее некоторую характеристику типа. Примерами атрибутов типов могут служить встроенный атрибут типа sizeof(T) в языке C++.

Концепция

Если F — функциональная процедура типа, то Arity(F) возвращает количество ее входных объектов.

Концепция

Функция типа — это отображение из типа в другой тип.
Примером определения функции типа является следующее: если задан ‘указатель на T ’, тип T .

Концепция

Если F — тип функциональной процедуры, то функция типа Codomain(F) возвращает тип результата. Если F — тип функциональной процедуры и i $<$ Arity(F), то индексированная функция типа InputType(F, i) возвращает тип i-го параметра

Концепция

Конструктор типа представляет собой механизм создания новых типов исходя из одного или нескольких существующих типов.

Концепция

Например, pointer(T) — встроенный конструктор типа, который принимает тип T и возвращает тип ‘указатель на T’; struct — встроенный n - арный конструктор типа; шаблон структуры — определяемый пользователем n - арный конструктор типа.

Концепция

Если T представляет собой n - арный конструктор типа, мы обычно обозначаем результат его применения к типам T_0 ,.. , T_{n-1} как $T_{T_0,..,T_{n-1}}$. Важным примером является *pair*, который, будучи применен к регулярным типам T_0, и T_1 , возвращает структуру struct типа $pair_{T_0,T_1}$ с элементом $m0$ типа T_0 и элементом $m1$ типа T_1 .

Концепция

Несколько более формально концепция представляет собой описание требований к одному или нескольким типам, выраженных в терминах существования и свойств процедур, атрибутов типов и функций типа, определенных на типах.

Концепция

Мы говорим, что концепция **моделируется** конкретными типами или что типы моделируют концепцию, если удовлетворяются требования к этим типам.

Концепция

Чтобы указать, что концепция C моделируется типами T_0 , .. , T_{n-1} , пишем $C(T_0, .., T_{n-1})$.

Концепция

Концепция C' **уточняет** концепцию C, если каждый раз, когда C' удовлетворяется для множества типов, C также удовлетворяется для этих типов. Мы говорим, что концепция C ослабляет концепцию C' , если C' уточняет C.

Концепция

Концепция типа представляет собой концепцию, определенную на одном типе.

Концепция

Например, в языке C++ определена концепция типа целочисленный тип, уточнениями который являются концепции типа целочисленный тип без знака и целочисленный тип со знаком, тогда как в STL определена концепция типа последовательность.

Концепция

$$C(T_0, .., T_{n-1}) \triangleq E_0 \wedge E_1 \wedge E_2$$

$\triangleq$ читается как “равно по определению”, T_i представляют собой формальные параметры типа, а E_j являются предложениями концепции, которые принимают одну из трех форм.

Концепция

1. Приложение предварительно определенной концепции, указывающее подмножество параметров типа, которое моделирует концепцию.
2. Сигнатура атрибута типа, функции типа или процедуры, которая должна существовать для любых типов, моделирующих концепцию. Сигнатура процедуры принимает форму $f : T \to T'$, где T домен; T' кодомен. Сигнатура функции типа принимает форму $F : C \to C'$, где домен и кодомен — концепции.
3. Аксиома, выраженная в терминах этих атрибутов типа, функций типа и процедур.

Концепция

Например, следующая концепция описывает унарную функциональную процедуру:

$$
\begin{aligned}
\mathit{UnaryFunction}(F) \triangleq\ & \\
& \wedge \mathit{FunctionalProcedure}(F) \\
& \wedge \mathit{Arity}(F) = 1 \\
& \wedge \mathit{Domain} : \mathit{UnaryFunction} \to \mathit{Regular} \\
& \quad F \to \mathit{InputType}(F, 0)
\end{aligned}
$$