**Содержание**

1. Введение

2. Философия языка

3. Основные термины и определения

4. Организация вычислительного процесса

5. Представление систем и типы данных

6. Основные управляющие структуры

7. Функции

8. Независимость семантики от синтаксиса

9. Расширение систем

**1. Введение**

На текущий момент существует много языков программирования, в тоже время постоянное появление новых говорит о неудовлетворенности программистов имеющимися инструментами. В связи с данным обстоятельством предлагается новый язык программирования, имеющий следующие особенности:

- Системный подход как универсальное средство описания и решения проблем. Целью такого подхода является представление единой платформы с минимально возможным количеством частных случаев и специфических механизмов. Высокоуровневое представление рассчитано на использование готовых модулей и библиотек;

- Единое иерархическое представление процессов и данных. Является реализаций системного подхода, позволяющей рассматривать и данные, и вычислительный процесс с единых позиций, в виде иерархических структур. Большинство современных (также как и устаревших) языков программирования имеют специализацию в типах данных, что обуславливает применение к ним специализированных отдельных операций. Это производительно и делает простым реализацию таких инструментов, но затрудняет создание сложных программ как конечный результат работы программиста. Именно такая постановка проблемы привела к внедрению функциональных элементов в современные императивные языки программирования – обобщенное программирование тому пример.

- Строгие правила использования функций (нельзя изменять входящие параметры внутри функции, элементы контрактного программирования). В целях уменьшения сцепляемости кода, в языке отсутствуют замыкания и функциям запрещено изменять свои входящие параметры. Поскольку результат функции всегда является иерархией, у программиста есть возможность вернуть несколько результатов, объединив их в одну структуру. Для упрощения отладки и сопровождения сам код функции в реализации языка жестко ограничен количеством действительных строк (то есть содержащих исполняемые строки кода, без пустых строк, описания внутренних функций и комментариев) 40 строками (1-2 видимых экрана). Функции также содержат элементы контрактного программирования для контроля и фильтрации входящих параметров. Минимальная сцепляемость кода позволяет динамически проводить ограниченный обмен функциями между системами.

- Универсальный базовый динамический тип данных. С целью удобного и простого преобразования все типы данных образуются от единой универсальной иерархии, что позволяет упрощенную манипуляцию данными на низком уровне. Защиту от произвольного доступа предоставляется средствами контрактного программирования.

- Инструменты интернациональной разработки на уровне среды исполнения. Представляют собой возможность смены синтаксиса языка программирования и возможность перевода программ из одного синтаксиса в другой.

**2. Философия языка**

В основу языка программирования положены идеи системного анализа. Все является системами. Составление программы есть описание системы, состоящей из взаимосвязей других систем. Система имеет внешнее проявление (открытую часть) и самостоятельное внутренне состояние (закрытую часть). Представление систем в языке программирование представляется иерархически с помощью деревьев. Система состоит из реализации (выраженной в виде описания функций) и структуры (описывающей взаимосвязь и подчиненность подсистем). Программа это система, но в тоже время она является ветвью дерева (например, она ветвь в структуре интерпретатора, который является частью вычислительных процессов операционной системы и т.д.).

Вычислительный процесс описывается функциями, которые представлены в виде дерева, состоящего из листьев (представленных операторами) и ветвей (представленных блоками кода – вызовами функций, условиями и циклами).

Функция может иметь внутренние именованные функции, доступ к которым возможен только из данной функции. Все функции не могут изменять входящие параметры.

Программа динамически, во время выполнения, имеет возможность для изменения структуры и состава любой из систем, созданных программистом. Общая задача программиста достигать целей через построение систем и связей между ними.

Для наглядного представления, что такое системы можно рассмотреть следующую модель. Вводные данные: имеются сведения о чем-либо. Например, перед нами печатающее устройство МФУ (многофункциональное устройство). Оно имеет в своем составе узел печати, узел подачи бумаги и узел сканирования изображений. Что можно сказать о МФУ как о системе? Прежде всего, нужно учесть, что любое представление, любая абстракция это **не полное отражение представляемого объекта**. МФУ имеет гораздо больше элементов в своем составе, не были указаны, например, блок питания, корпус, индикаторы на передней панели и т.д. Но в модели нет необходимости представлять все, что входит в МФУ, так как **модель любого объекта содержит только те элементы, которые нужны для решения задачи**, ради которой и создается модель. Элементы, которые нужны для решения задачи определяются **создателем модели**. Принцип, по которому компоненты включаются в модель, называется Бритва Оккама (закон экономии), в общей форме он гласит: «Не следует привлекать новые сущности без крайней на то необходимости». Поэтому же принципу в МФУ включен именно узел сканирования изображений, а не источник света, зеркало, линза, матрица ПЗС, аналогово-цифровой преобразователь и сигнальный процессор, хотя если рассматривать МФУ вне данной модели представления сведений, все эти блоки там присутствуют. В некоторых случаях, все же отдельные компоненты рассматриваются и используются в модели, но прямое обращение к ним нужно не часто. В таких случаях, модель конкретизируют, путем перечисления **всех необходимых компонентов (и только необходимых!)** объекта нижнего уровня. Принцип включения компонентов нижнего уровня в логическое представление (то есть включение всех линз, матриц, зеркал и т.д. в узел сканирования изображений) называется **принципом композиции**. Он заключается в придании емкого простого термина сложным устройствам и явлениям. Например, никто не говорит – из-за горизонта планеты возникло оптическое представление звезды класса желтый карлик типа G2V средней плотности 1,4 г/см3 с эффективной температурой 5780 кельвин на 73 % массы состоящего из водорода и.т.д. Говорят: «Наступило утро» или «Солнце взошло».

Системы имеют два важных показателя – это **состав и структура**. Состав – те компоненты из чего состоит система. Снова нужно помнить о бритве Оккама, так как МФУ может состоять как из описанных выше узлов, так и из пластмассы, цветных металлов и редкоземельных элементов. Оба варианты фактически верны, если задачей ставится **исследование системы**. Состав системы напрямую зависит от **решаемой задачи, предназначения системы**.

Отсюда следует, что одни и те же реальные объекты могут быть представлены разными системами, в зависимости от того, с какой целью нужно представлять объект.

Структура системы это **взаимосвязи между компонентами, из которых состоит система**. Очевидно, что если поменять местами узел сканирования и узел подачи бумаги, то МФУ потеряет свою функциональность, несмотря на то, что **состав системы не изменился**. Важно не только из чего состоит система, но и **как составные части между собой взаимодействуют**. Груда деталей не является машиной. Если смешать все ингредиенты в одной чашке совсем не факт, что в результате получится бисквитный торт.

Образующим постулатом является **обязательное наличие состава и структуры системы**, даже если они на момент рассмотрения неизвестны (например, модель черный ящик).

**Все системы** обладают рядом свойств:

- Эмерджентность – это свойство системы получать новые свойства, отличные от тех, что обладают компоненты входящие в систему. Например, прутики веника легко сломать по одному, но сломать веник, состоящий из прутьев, уже гораздо сложней.

- Целостность – это проявление все того же принципа бритвы Оккама – каждый компонент системы определяет систему, ее поведение и свойства. Отсутствие любого из компонентов системы разрушает данную систему (хотя при этом можно говорить о наличии иной системы). Так для рассмотренных представлений о МФУ совершенно не важен цвет корпуса, но отсутствие узла печати приведет к потере МФУ некоторых своих свойств. Другими словами только определенный состав компонентов и определенный набор из взаимосвязей (структура системы) этих компонентов однозначно определяют, что перед нами именно описываемая система, а не иная.

- Организованность – обязательное наличие уникальной структуры системы, которая характеризует именно данную систему (так наличие в системе ракетного двигателя может нам гарантировать, что перед нами не МФУ, а нечто иное).

- Функциональность – проявление поведения системы, при взаимодействии с внешним миром (другими системами). Так с помощью МФУ можно делать печатные копии документов, осуществлять печать и т.д. При этом МФУ не готовит суши, не продает газеты и не стирает белье.

- Структурность – определенная структура системы, характеризующая именно данную систему. Например, МФУ может производить печать документов, и принтер может производить печать документов. Но это разные системы, поскольку их структура различна, несмотря на некоторые общие функции (и например общий цвет, одного производителя, одного формата бумаги, одинакового тонера, картриджа и т.д.).

- устойчивость – это возможность сопротивляться внешним воздействиям и оставаться системой. Например, падение с небольшой высоты или слабый удар МФУ способно выдержать без потери своей прежней функциональности. МФУ способно работать также при некоторых отклонений параметров электрического тока (например, и при напряжении 230В и при 180В, вместо положенных 220В). При этом устойчивость ограничена некоторыми рамками, так входное напряжение свыше 1000В сделает МФУ неработоспособным, вызвав изменение свойств компонентов и взаимосвязей между ними.

- надежность – возможность системы сохранять некоторую работоспособность и функциональность при некоторых повреждениях ее компонентов. Например, повреждения корпуса необязательно приведут к отказу компонентов МФУ и полной потери работоспособности. При этом с точки зрения сравнения с эталонными представлениями о МФУ, это уже иная система с иными свойствами (например, у нее может измениться устойчивость внешним воздействиям).

- адаптируемость – способность систем изменять свои свойства под воздействием окружающей среды, сохраняя при этом свою основную функциональность и предназначение.

Ввиду наличия большого количества имеющихся сведений, ярким примером систем являются живые организмы, например, одноклеточные организмы. Все свойства систем в одноклеточных организмах выражены наиболее отчетливо и к тому же подробно изучены и подтверждены.

Все вышеописанные концепции, принципы и свойства применимы к программам и их компонентам, как к системам, разрабатываемым программистами.

**3. Основные термины и определения**

В данном разделе будет представлена предметная область, в рамках которой, предлагается создание и использование программ на языке программирования Валентина-2. Дальнейшее описание языка будет построено на данной трактовке терминов (которые, возможно, будут отличаться от общепринятых представлений).

**Атомарный тип данных** – тип данных, значения которых считаются неделимыми. Например, в большинстве языков программирования цифра – атомарный тип данных. В языке программирования Валентина-2 атомарный тип данных – строковой символ. Так как числа хранятся в виде строк – цифра, в данном случае, является абстракцией, ограниченным подмножеством значений, заключенных в строковой символ.

**Абстрактный тип данных** – тип данных, работа с которым не зависит от конкретных значений, содержащихся в структуре. Например, стек в абстрактной модели может содержать любые типы, в качестве значений, хранящихся в стеке – числа, строки, вектора, записи и т.д. В конкретной реализации стека (например, в аппаратной реализации в микропроцессорах) данные, как правило, ограничены конкретным типом.

**Абстракция** – мысленное представление, обобщение свойств систем. Например, машина это абстракция, собирательный образ. Конкретный автомобиль или компьютер (и то и другое машина) является объектом реального мира.

**Базовый тип данных** – тип данных, на основе которого построен другой. В языке программирования Валентина-2 строковой тип является базовым для чисел. Все остальные типы (а также доступ к строкам и числам) реализуется через тип данных «система».

**Ветвь** – в иерархических типах данных любой элемент иерархии.

**Динамический тип данных** – тип, природа и/или состав данных которого могут меняться во время выполнения программы.

**Иерархия** – структура, в которой данные низшего звена подчинены высшему звену. В программировании иерархии часто реализуются через тип данных Дерево.

**Полный сцепленный ключ системы (ключ системы)** – указатель на систему, состоящий из имен всех надсистем, которые нужно пройти от корня до искомой системы. Может использоваться также **локальный сцепленный ключ системы**, когда путь до искомой системы отсчитывается от текущей системы, а не от корневого узла.

**Комментарий** – пояснения в тексте программы, как правило, не влияют на исполнение программы. Могут быть различные типы комментариев – для программистов, для автоматического документирования и т.д.

**Корневой узел** – в типе данных дерево, центральный узел, не имеющий родителя.

**Модель** – система с отражением базовых характеристик объекта, важных для исследования или решения задачи. Какие именно характеристики, свойства включаются в модель, определяется разработчиком модели.

**Модуль** – логически законченная программная единица. Предназначен для структурированная программ.

**Наследование** – концепция программирования, по которой объект может получать свойства другого объекта (или описания объекта).

**Надсистема** – система, в которую входит объект.

**Подсистема** – составной элемент рассматриваемой системы, часть системы, объект системы, подсистема обязательно входит в структуру системы и имеет связи с другими подсистемами.

**Простая система** – в языке программирования Валентина-2 система, имеющая только имя и строковое значение. Является отражением простых переменных в других языках программирования. Простая система, с точки зрения программиста, ни чем не отличается от остальных систем (но имеет особенности реализации, ускоряющие работу с ней).

**Система** – форма представления объекта (надсистема) как группы объектов (подсистемы), находящихся во взаимосвязях друг с другом.

**Сложная система** – в языке программирования Валентина-2 системы, имеющие сложную природу (например, имеют в своем составе функции, закрытую часть и т.д.). В других языках программирования наиболее близко соответствует термину «объект».

**Структура системы** – описание взаимосвязей объектов внутри системы.

**Состав системы** – множество объектов (подсистем) входящих в систему.

**Сцепляемость кода** – характеристика фрагмента исходного кода, отражающего сложность переноса данного фрагмента из одной структурной единицы кода (функция, модуль, программа) в другую. Например, перенос функции, имеющей замыкание, требует дополнительного анализа всех читаемых/изменяемых объектов, объявленных вне функции. Сцепляемость кода такой функции является высокой и является барьером при переносе кода, рефакторинге и сопровождении программы. Наименьшую сцепляемость кода имеют механизмы, встроенные в язык программирования, например, встроенная функция языка, в большинстве случаев может быть вызвана из любого места программы на том же языке программирования.

**Реализация функций системы** – в языке программирования Валентина-2 специальная секция в описании системы (в исходном коде), в которой описывается код функций, входящих в систему.

**Родитель** – в иерархиях, элемент, вышестоящий в иерархии по отношению к рассматриваемому узлу.

**Тип данных** – множество всех возможных значений, которые может принимать структура. Часто в тип данных неявно включают еще и функции (набор операций), обеспечивающие работу с типом данных.

**Тип системы** – в языке программирования Валентина-2 специальное строковое значение, служащее для целей контрактного программирования. В частности по типу системы функции могут контролировать и/или фильтровать входящие параметры. В простой системе тип системы – пустое строковое значение (в сложной системе может быть любой тип системы, включая пустое строковое значение).

**Узел** – любой отдельный элемент иерархии.

**Условия** – одна из форм ветвления, способ организации вычислительного процесса, в котором выполнение части программы возможно при соблюдении определенного условия.

**Функция** – логически законченный фрагмент программы, к которому можно обращаться из других участков программы (в том числе и из самой функции). Особенностью функции является возможность передачи в часть программы, вызвавшей функцию, какого либо значения, результата.

**Циклы** – способ организации вычислительного процесса, в котором выполнение части программы возможно многократно при соблюдении определенных условий.

**4. Организация вычислительного процесса**

Вычислительный процесс в языке программирования является императивным – программа это описания систем и набор инструкций, который исполняется виртуальной машиной, создающей модель языка программирования Валентина и реализующей его семантику. Программа исполняется в изолированном окружении и не имеет прямого доступа к ресурсам компьютера. Вычисления и хранение информации проводится в модели, оторванной от аппаратных особенностей компьютера. Наиболее близкой парадигмой программирования является объектно-ориентированное программирование, в тоже время имеются некоторые небольшие отличия. Для того чтобы не возникала путаница с общепринятыми представлениями в языке программирования используется собственная терминология, основанная на положениях системного анализа.

Элементы, с помощью которых, осуществляется организация вычислительного процесса:

- ветвление;

- циклы;

- подпрограммы;

- логическое представление компонентов программы в виде систем.

Ветвление представляет собой способ управления вычислительным процессом с помощью исполнения набора инструкций (операторов) при наступлении определенных условий либо при определенном результате выражения. Прямой безусловный переход (аналог goto в других языках программирования) отсутствует, так как не соответствует иерархическому способу представления моделей. В языке программирования ветвление реализовано с помощью условий двух типов:

- простое условие – выполнение набора инструкций (операторов) при соблюдении определенного логического условия;

- условия с двумя ветвями вычислений – выполнение одного набора инструкций (операторов) при соблюдении логического условия и выполнение иного набора инструкций (операторов) при не соблюдении логического условия.

- подпрограммы – участок программного кода, выполняющий последовательность действий и имеющий возможность многократного использования. В языке программирования Валентина подпрограммы представлены функциями. Функция это подпрограмма, имеющая имя и параметры. Более подробное описание имеется в разделе 7 Функции.

- логическое представление компонентов программ в виде систем – описание частей программы в виде определенной структуры, позволяющей разбивать программу на логические части.

В связи с особенностями вычислительной модели язык имеет небольшой количество операторов, большая часть из которых предназначена для управления вычислительным процессом. Остальные возможности реализованы посредством встроенных функций.

**5. Представление систем и типы данных**

Система центральное понятие языка программирования Валентина. Все является системами. Язык предусматривает описание систем на этапе проектирования и создания программы, а также имеет возможности по изменению структуры системы во время исполнения.

Общий вид системы:

**Система:** Тип\_системы

**Внешние системы:** Список\_имен\_внешних\_систем\_через\_запятую

**Точка входа:** Имя\_функции\_для\_запуска\_системы\_как\_самостоятельной\_программы

**Конструктор:** Имя\_функции\_ответственной\_за\_подготовку\_системы\_к\_работе

**Интернациональные описания:**

Идентификатор\_на\_национальном\_языке=Идентификатор\_на\_английском

**Структура**

**Открытая часть**

Список\_и/или\_описания\_систем

Список\_функций

Значение

**Закрытая часть**

Список\_и/или\_описания\_систем

Список\_функций

Значение

**Реализация функций**

Список\_описаний\_функций

**Система закончена**

Системы могут иметь и краткую, простую структуру, описание которых возможно неполным использованием секций. Так если в системе отсутствуют функции, то секция реализация функций может отсутствовать.

Системы в некотором смысле соответствуют объектам, а их описания классам. Но имеются и отличия, так в объектах затруднено, а часто и вовсе невозможно изменение структуры – если объект имеет ссылку на какой либо тип данных, он всегда будет иметь эту ссылку в своем составе. Если запись (например, struct в С++ или record в Pascal) содержит в своем составе целое число, она всегда будет его содержать. Структура таких объектов неизменна (но могут менять значения объектов или, например, числовых типов). В некоторых языках программирования допускается хранить указатели, а не данные. Возможно, менять значения указателя, чтобы он указывал на другой тип данных, но сам указатель удалить из объекта нельзя. Можно создать, например, динамический массив указателей и манипулировать указателями, но тогда нельзя будет удалить массив указателей.

Системы могут быть изменены как в значениях (данные в виде строковых значений и функций) так и в структуре (метаданные о составе и подчиненности подсистем). Единый подход к элементам системы делает возможным простое управление не зависимо от того, какая структура данных реализована посредством систем.

Также особенностью систем является прямое копирование для получения идентичной копии системы (наследование не от описания системы, а от действующего экземпляра), включая структуру и всю текущую информацию, содержащуюся в системе. Прямое копирование означает получение полной копии объекта, а не указателя на объект, существующей в единственном экземпляре.

В языке программирования Валентина имеется два основных типа данных – строки и системы. Все остальные типы и данные порождены использованием двух основных. Строки используются для хранения данных, системы для организации данных и манипуляции с метаданными.

Строки являются атомарным и базовым типом данных. Системы являются базовым типом данных. Так как роль систем в программировании на языке Валентина велика (фактически программирование это есть описание системы), то и множество возможных порождаемых систем велико (но предположительно конечно и определяется конкретной реализацией).

Для возможности идентификации системы имеет собственную характеристику, также называемой типом. Чтобы избежать путаницы при одновременном использовании типа и типа системы, последний допускается именовать природой системы. Тип системы это строковое значение, предназначенное для использования программистом для описания системы. Тип системы динамически изменяемая неавтоматическая характеристика системы. Динамически изменяемая характеристика означает, что во время исполнения программы допускается изменение типа системы. Неавтоматическая характеристика означает, что тип системы не отражает реального содержания системы и полностью управляется программистом. Это важное отличие от типов объектов в иных языках программирования, где тип жестко отслеживается и определяется системой исполнения даже в языках с мягкой типизацией. Тип системы не является автоматическим описанием подмножества систем, которой принадлежит данная система (так как все системы принадлежат общему описанию систем). Тип системы предназначен для целей контрактного программирования и может отслеживаться с помощью специальных инструментов функций (подробней об этом в разделе с описанием функций).

Для возможности использования понятия «переменная» принятого в математике и классическом структурном программировании системы условно можно разделить на два вида: простые и сложные. Простые системы имеют только имя и значение (тип системы пустая строка), сложные системы все остальные. Доступ к содержимому простой системы возможен по упрощенному варианту, например, а=5. Доступ к сложной системе требует указания подсистемы, к которой относятся данные, например, машина.двигатель.мощность=110, либо использования встроенных функций языка для работы с системами.

Согласно представляемой модели данных все системы образуют от одного общего предка, недоступного программисту. Условно он называется корень (условно, потому что в данной модели вычислений корень всего лишь ветвь в более сложно организованной системе – виртуальной машине, которая всего лишь ветвь в более сложной системе – интерпретатор и т.д.). Для обеспечения функциональности Корень имеет еще ряд ветвей недоступных для программиста – это ветвь описаний систем и ветвь параллельных систем. Ветвь описания систем содержит все системы, описания которых встречаются в программе. Программист не может менять системы, располагающиеся там, но может копировать системы из данной ветви. Ветвь параллельных систем аналогичным образом содержит все описания систем, физически расположенных в иных модулях программы, но требуемых для работы текущей системы. В случае если в момент исполнения программы требуется создать систему из описания, сначала проводится поиск систем в ветви описаний системы (то есть описания систем из данного исходного текста), затем в ветви параллельных систем (то есть описания систем из внешних исходных текстов).

**6. Основные управляющие структуры**

Основные управляющие структуры языка:

1. Условный оператор. Специальная конструкция языка программирования, предназначенная для исполнения участка программы при наступлении определенного условия.

Условие состоит из следующих элементов – условия исполнения, тело условия. Условие исполнения – логическое выражение, истинность которого управляет вычислительным процессом. Тело условия – участок или несколько участков программы, выполнение которых возможно в зависимости от вычисленного логического выражения.

а) Простое условие с одной ветвью. Это условный оператор, в котором участок программы будет исполнен, в случае если логическое выражение, заданное в условии исполнения, является истинным.

б) Условие с истинной и ложной ветвями. Это условный оператор, в котором имеется два участка программы, выполнение которых зависит от истинности условия исполнения (в случае истинности выполняется один участок, называемый истинной ветвью, в случае ложности другой, называемый ложной ветвью).

Условие является частной формой ветвления.

2. Блок обработки исключений. Специальная часть программы, состоящая из условия наступления исключения, участка программы для которого отслеживается исключение и участка программы, выполняющегося в случае возникновения условия наступления исключения. Блок обработки исключений предназначен для реакции программы на ошибки, возникающие во время выполнения.

3. Циклы.

Циклы являются управляющей конструкцией языка программирования, предназначенные для многократного исполнения участка программы. Цикл имеет в своем составе несколько элементов:

- тело цикла – последовательность команд, предназначенных для многократного выполнения в цикле;

- управляющее условие – условие выполнение (или невыполнение) которого приводит к повторному выполнению тела цикла;

- шаг цикла – однократное исполнение тела цикла;

Цикл является частной формой ветвления.

В языке Валентина имеется несколько видов циклов.

а) Цикл-счетчик. Цикл, в котором имеется специальная переменная-счетчик шагов цикла. Условием работы цикла является достижением переменной-счетчиком какого-либо значения, изменение переменной в цикле происходит автоматически.

б) Цикл с предусловием. Цикл, в котором выполнение тела цикла зависит от определенного логического условия, которое рассматривается каждый раз перед следующим шагом цикла.

в) Цикл перебора подсистем. Цикл, в котором имеется специальная переменная, получающая каждый раз одну из подсистем указанной в цикле системы.

Вычислительный процесс в программе представлен как обход дерева операторов языка программирования. Поэтому все управляющие структуры это ветви иерархии, часть из которых может иметь свои узлы. Так, например, условие может иметь одну истинную ветвь или одну истинную и одну ложную ветвь.

**7. Функции**

Функция это логически законченный фрагмент программы, к которому можно обращаться из других участков программы (в том числе и из самой функции). Учитывая отсутствие контроля типов систем, при проектировании функций используются элементы контрактного программирования. Контрактное программирование предполагает контроль поведения программы и ее компонентов, для чего в язык программирования вводятся специальные механизмы и условия. В языке программирования Валентина имеются только базовые возможности контрактного программирования – предусловия и постусловия, выражаемые в **спецификаторах функций**.

Модель вычислений, построенная на системах, предполагает использование систем в качестве параметров и результатов функций. Так как функция это тоже система, следовательно, она может быть передана в качестве параметра и возвращена как результат функции. Соответственно, все функции изначально являются функциями высшего порядка, без внедрения дополнительных концепций и специальных механизмов работы с функциями. Также при проектировании программ следует учитывать, что к функциям применимы операции над системами.

**7.1. Спецификаторы функций**

Спецификаторы функций это специальные методы контроля обрабатываемой информации, бывают двух типов – предусловия, проверяют входящие параметры и постусловия, проверяют результат функции. Например, спецификатор может сверять систему-результат функции со списком доступных типов (в случае несоответствия будет генерироваться исключение).

Предусловия контролируют диапазон значений (значений системы, например, тип системы это тоже значение) входящих параметров (например, можно контролировать ситуацию, когда делитель в операции деления был не равен нулю). Постусловия отслеживают по выходу из функции ее результат на соответствие определенным условиям. Сами спецификаторы функциям не принадлежат и накладывают ограничения в момент вызова функции. Основная задача спецификаторов – ускорение отладки программы, второстепенная предоставление внешним средствам возможности автоматического документирования параметров функций. Главным отличием от простой генерации исключительной ситуации (например, при несоответствии типов) является возможность отслеживания логических ошибок за счет формализации условий передачи параметров. Спецификаторы могут ограничивать диапазон значений параметров, даже тогда когда значение параметра не является основанием для генерации исключительной ситуации (например, число ноль, допустимое значение для целых чисел, но может приводить к операции деления на ноль, соответственно можно отслеживать, чтобы в функцию никогда не передавался ноль). Сами спецификаторы представляют собой набор логических условий, которым должны соответствовать параметры в момент вызова и выхода из функции. Несоблюдение любого из условия приведет к генерации исключения. Таким образом, возможно, накладывать на параметры ограничения разной природы (отслеживать диапазон значений, проверять наличие подсистем, контролировать тип и т.д.).

**7.2. Внутренние (вложенные) функции**

Как уже отмечалось выше, язык программирования не имеет анонимных функций, каждая функция должна быть именована. Это следует из свойств систем и операций над ними – функция система, а значит, к ней должны быть применимы, если не все операции, то хотя бы базовые – создание, копирование, перемещение и т.д. И, следовательно, необходима возможность адресации к функции. Так как функция система, она имеет возможность хранить внутри себя подсистемы и иметь подсистемы-функции. Это не специальная возможность, а функциональность, представленная базовыми свойствами модели вычислений. Таким образом, в рамках общей модели, программисту предоставляется возможность логически организовывать код. Кроме того, спецификация языка предполагает ограничение на размер действительных исполняемых строк в функции. Это требование предназначено для принуждения программиста к логической декомпозиции задачи.

**7.3. Встроенные функции**

Встроенные функции это механизм расширения языка программирования Валентина. Они изначально описаны при реализации языка разработчиком и доступны сразу, их описание не требуется в системе. Встроенные функции реализуют возможности по управлению системами и их компонентами, работой со строками и т.д., то есть предоставляют базовый набор для построения более высокоуровневых систем. Поскольку они реализованы в рамках общей модели, на их вызов допускается накладывать ограничения с помощью спецификаторов (при этом ограничения на их результат не имеют смысла, так как программист не имеет возможности исправить поведение встроенных функций). Дальнейшее развитие и добавление новых возможностей языка предполагается путем введения дополнительных встроенных функций.

Представления о встроенных функциях согласуются с общей моделью следующим образом. Программа является подсистемой системы более высокого уровня. При этом назначение и полная функциональность надсистемы неизвестна (предполагается, что эта надсистема интерпретатор, но в данном случае это неизвестно). Программа это такая же ветвь вычислений, как и все ее подсистемы. При этом программа может представляться как система-функция, которая имеет входящие параметры-аргументы и исходящий параметр-результат. Среди прочих аргументов в программу также передаются и встроенные функции (так как в рамках модели все функции являются функциями высших порядков). Таким образом, помимо описанных в программе функций, программа получает и дополнительные возможности в виде встроенных функций.

**7.4. Классификация встроенных функций**

Встроенные функции условно подразделяются на:

1) Функции для работы со строками:

- определение длины строки;

- поиск подстроки;

- копирование части строки;

- приведение строки к верхнему или нижнему регистру.

2) Функции для изменения структуры системы:

- удаление подсистемы;

- добавление подсистемы;

- копирование подсистемы;

- проверка существования указанной подсистемы;

- определение количества подсистем;

3) Прочие функции.

**8. Независимость семантики от синтаксиса**

Особенностью языка программирования Валентина является незакрепленный синтаксис. Это значит, что программист может использовать свой синтаксис, доступный для изменения в некоторых пределах. Поскольку синтаксис меняет только внешний вид исходного текста программы, но не влияет возможности интерпретатора и его семантику, в терминах языка программирования Валентина он называется **диалектом**.

По умолчанию используются два стандартных диалекта. Первый предназначен для чтения исходных текстов и представляет собой многословную запись, приближенную к естественной записи. Второй диалект предназначен для быстрой записи программ и потому ориентирован на использование кратких ключевых слов и использование спецзнаков. Интерпретатор имеет встроенные средства для перевода текста программы из одного диалекта в другой.

Для описания конструкций операторов в диалектах используется встроенный метаязык, представляющий простую форму описаний. Описание диалекта осуществляется в виде обычного текстового файла, состоящего из секций, которые содержат наборы вида «ключ=значение»:

- General. Это секция общих настроек диалекта языка. Данная секция включена по умолчанию (это значит, что при разборе файла, содержащего описание диалекта, считается, что включена секция General и ее можно не объявлять, если сразу описываются общие настройки диалекта). General может содержать следующие ключи:

1) Name – имя набора;

2) Remark – описание набора;

3) OpenQoute – открывающая кавычка для описания текстовых констант;

4) CloseQoute – закрывающая кавычка для описания текстовых констант;

5) OpenBracket – открывающая скобка (используется при объявлении и вызове функций);

6) CloseBracket – закрывающая скобка (используется при объявлении и вызове функций);

7) Divider – разделитель, используемый при перечислении параметров;

8) Token – лексема, которая в сплошном тексте должна рассматриваться как отдельный элемент.

**ТУТ ПРОДОЛЖИТЬ ПРО ОПИСАНИЕ СОДЕРЖИМОГО ФАЙЛА ДИАЛЕКТА!!!**

Несмотря на возможность некоторой свободы в описании собственного синтаксиса, отличного от стандарта, имеются и правила общие для всех программ языка Валентина.

Первым в описании следует обозначение имени системы.

9. Расширение систем