и ничего более), то есть зарезервированные слова нельзя использовать в качестве имен переменных пользователя.

Знаки операций – это один или несколько символов, определяющих действие над операндами. Внутри знака операции не может быть пробелов (пробел – это всегда разделитель). Например, в выражении x+y знак «+» означает операцию сложения, а x и y являются операндами.

Операнд – это константа, переменная или вызов метода (функции). Операнды, связанные знаками операций, образуют выражения. Тип выражения определяется типом операндов.

Литералы (константы) – это величины, которые неизменны. Компилятор определяет тип константы по ее внешнему виду. Литералы в языке C++ могут быть целые, вещественные, символьные и строковые.

Разделитель используются для разделения или для группировки элементов. Примеры разделителей: пробелы, скобки, точка, запятая.

Лексемам в языке человека соответствует понятие слово. В литературе, посвященной трансляции с языков программирования, часто используется термин токен, имеющий тот же смысл. Почти все типы лексем (кроме ключевых слов и идентификаторов) имеют собственные правила словообразования, включая собственные подмножества алфавита.

УДК 381

Шот А.В.

СОВРЕМЕННЫЕ ПАРАДИГМЫ ПРОГРАММИРОВАНИЯ

БНТУ, Минск
Научный руководитель: Дробыш А.А.

Парадигма программирования («пример, модель, образец») – исходная концептуальная схема постановки задач и их решения

вместе с языком, ее формализующим, и формирующая стиль программирования.

Основные стили, или парадигмы программирования, к которым обычно относят императивное, функциональное, логическое и объектно-ориентированное программирование возникли более сорока лет назад вместе с первыми языками программирования и развивались сначала относительно независимо друг от друга. Каждая из парадигм отображает определенную модель вычислений, включая структуры данных и механизмы управления, и с ней связан определенный класс прикладных задач, которые удобно решать средствами данной парадигмы. В настоящий момент большинство современных языков программирования обычно включают средства и приёмы программирования различных парадигм, хотя классифицируются согласно средствам своего ядра.

Парадигмы программирования

Императивная, или процедурная парадигма программирования является наиболее известной. Она развилась на базе низкоуровневых языков (машинные коды, ассемблер), основанных на архитектуре фон Неймана. Императивная программа состоит из последовательно выполняемых команд и вызовов процедур, которые обрабатывают данные и изменяют значения переменных программы. Переменные при этом рассматриваются как некоторые контейнеры для данных, подобно ячейкам памяти компьютера.

Функциональная парадигма программирования является менее традиционной, и в тоже время более древней, поскольку получила развитие из вычислений по алгебраическим формулам. Функциональная программа состоит из набора взаимосвязанных и, как правило, рекурсивных функций. Каждая функция определяется выражением, которое задает правило вычисления её значения в зависимости от значений ее аргументов. Выполнение

функциональной программы заключается в последовательном вычислении значений функциональных вызовов.

В менее традиционной и необычной логической парадигме программа рассматривается как множество логических формул: аксиом (фактов и правил), описывающих свойства некоторых объектов, и теоремы, которую необходимо доказать. В свою очередь, выполнение программы – это доказательство теоремы, в ходе которого строится объект с описанными свойствами.

Основные различия указанных парадигм касаются не только концепции программы, но и роли переменной. В отличие от императивных программ, в функциональных и логических программах отсутствует явное присваивание значений переменным и, как следствие, побочные эффекты. Переменные в таких программах подобны переменным в математике: они являются обозначением функциональных аргументов или объектов, конструируемых в процессе доказательства. Еще одна яркая особенность функциональной и логической парадигм – использование рекурсии вместо циклов.

В получающей все большее распространение объектно-ориентированной парадигме программа описывает структуру и поведение вычисляемых объектов и классов объектов. Объект обычно включает некоторые данные (состояние объекта) и операции с этими данными (методы), описывающие поведение объекта. Классы представляют множество объектов со схожей структурой и схожим поведением. Обычно описание классов имеет иерархическую структуру, включающую полиморфизм операций. Выполнение объектно-ориентированной программы представляет собой обмен сообщениями между объектами, в результате которого они меняют свои состояния. Характерные свойства основных парадигм программирования представлены Таблице.

Таблица – Свойства парадигм программирования

Парадигма	Ключевой концепт	Программа	Выполнение программы	Результат
Императивная	Команда	Последовательность команд	Исполнение команд	Итоговое состояние памяти
Функциональная	Функция	Набор функций	Вычисление функций	Значение главной функции
Логическая	Предикат	Логические формулы	Логическое доказательство	Результат доказательства
Объектно-ориентированная	Объект	Набор классов объектов	Обмен сообщениями между объектами	Результирующее состояние объектов

Парадигмы и языки программирования

Большинство современных языков аккумулируют в себе элементы и приемы нескольких стилей, и тем не менее их можно классифицировать по основному их ядру, реализующему приёмы определенной парадигмы программирования, в частности:

Императивная парадигма: языки Паскаль, Си, Ада;

Функциональная парадигма: языки Лисп, Рефал, Плэнер, Schema, Haskel;

Логическая парадигма: языки Пролог и Datalog;

Объектно-ориентированная парадигма: Smalltalk, Eiffel.

В целом, сравнивая разные языки и парадигмы, следует отметить существенное отличие средств и приёмов традиционной императивной парадигмы программирования от средств и приёмов нетрадиционных парадигм – функциональной и логической. Языки программирования, основанные на нетрадиционных парадигмах, отличаются ещё и методом реализации: программы на таких языках обычно интерпретируются,

а не компилируются, как в императивных и императивных объектно-ориентированных языках. Еще одно важное их отличие – они ориентированы на символьную обработку данных.

Поскольку не существует языка, в полной мере содержащего возможности всех парадигм, для их изучения необходимо подобрать несколько различных языков программирования.

Важность изучения именно парадигм программирования обосновывается также тем, что будущее современных популярных языков программирования нам неизвестно. История развития информатики и программирования показывает, что некоторые языки оказались мертворожденными, другие быстро потеряли свою популярность, и подобная участь может постигнуть некоторые из современных языков. В тоже время основные парадигмы программирования остаются неизменными, как и их базовые средства и приемы. Таким образом, изучение парадигм является важной компонентой обучения специалистов в области информатики и программирования. Основные парадигмы программирования существенно отличаются по своим приемам и средствам, а поэтому имеют различные области применения в практике программирования.

УДК 381

Юрьев В.А.

ИСПОЛЬЗОВАНИЕ УКАЗАТЕЛЕЙ В C/C++

БНТУ, Минск
Научный руководитель: Дробыш А.А.

Указатель – переменная, значением которой является адрес ячейки памяти. То есть указатель ссылается на блок данных из области памяти, причём на самое его начало. Указатель может ссылаться на переменную или функцию. Для этого нужно знать адрес переменной или функции. Так вот, чтобы узнать адрес конкретной переменной в C++ существует