2. Прямое управление динамической памятью, что опять же важно для скорости работы.

Недостатки:

1. Трудный для понимания синтаксис, который вызывает ошибки. Близкие по обозначению разнонаправленные операции (можно спутать).

2. 3 основных операции (присваивание, инкрементация, декрементация) в сочетании с другими операциями предполагает создание сложных для чтения выражений (усложняет диагностику).

3. Необходимо очень внимательно работать с указателями, ссылками, динамической памятью, проверять не происходит ли выход за границы массива.

УДК 372

Чайко Е.Ю.

АЛФАВИТ ЯЗЫКА ПРОГРАММИРОВАНИЯ C++

БНТУ, Минск
Научный руководитель: Дробыш А.А.

Алфавит (или множество литер) языка программирования C++ основывается на множестве символов таблицы кодов ASCII. Алфавит C++ включает:

– строчные и прописные буквы латинского алфавита (мы их будем называть буквами),

– цифры от 0 до 9 (назовём их буквами-цифрами),

– символ '_' (подчерк – также считается буквой),

– набор специальных символов: " { } , | [ ] + - % / \ ; ' : ? < > = ! & # ~ ^ . * и прочие символы.

В тесте программы можно использовать комментарии. Комментарий – это последовательность символов, которая игнорируется компилятором языка C++. Комментарий имеет следующий вид: /*<символы>*/. Комментарии могут

занимать несколько строк, но не могут быть вложенными. Кроме того, часть строки, следующая за символами //, также рассматривается как комментарий. Компилятор не понимает смысл комментариев, поэтому не существует способа проверить, что комментарий содержателен, имеет какое-то отношение к программе и не устарел. Написание «правильных» комментариев может оказаться не менее сложной задачей, чем написание самой программы. При неправильном использовании комментариев читабельность программы может серьезно пострадать.

Алфавит C++ служит для построения слов, которые в C++ называются лексемами. Лексема — это последовательность из одного или несколько символов, представляющая определенный смысл. Границы лексем определяются другими лексемами, такими, как разделители или знаки операций. Этой же цели служит множество пробельных символов, к числу которых относятся пробел, символы горизонтальной и вертикальной табуляции, символ новой строки, перевода формата и комментарии.

Существует несколько видов лексем: идентификаторы (имена объектов); ключевые (зарезервированные, служебные) слова; знаки операций; литералы; разделители;

Идентификатор – это последовательность букв, цифр и символов подчеркивания. Идентификаторы используются для обозначения имен объектов, используемых в программе. Всё, что применяется в программе, имеет имя. Имена (названия) имеют константы, переменные, методы, классы и т.д. Имя не может начинаться с цифры. Длина имени произвольная. В идентификаторах допускается применять национальные шрифты, например, русские буквы, но это нежелательно.

Ключевые слова – это служебные слова, которые зарезервированы в языке, их можно использовать только по прямому назначению (например, for – это заголовок оператора цикла

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