МИНИСТЕРСТВО ВЫСШЕГО И СРЕДНЕГО ОБРАЗОВАНИЯ РЕСПУБЛИКИ УЗБЕКИСТАН

ТАШКЕНТСКИЙ ИНСТИТУТ ТЕКСТИЛЬНОЙ И ЛЕГКОЙ ПРОМЫШЛЕННОСТИ

Кафедра: «Информатика и ИТ»

Курсовая работа

на тему:

«Основы алгоритмического языка C++»

Выполнила студентка группы 19р-15

Сабитова Мадина

Принял: Абдурахманов А.

Ташкент 2016

Содержание

Введение

Сегодня, как и во всем мире, на приоритетные позиции в нашей республике выходят компьютерные и информационные технологии, развитие и модернизация сетей телекоммуникаций и передачи данных, доступа к интернету.

В мае 2001 года Президент страны, выступая на пятой сессии ОлийМажлиса, поставил конкретные задачи по достижению Узбекистаном высокого уровня внедрения информационных технологий в производство, образование, повседневную жизнь людей.

30 мая 2002 года был подписан Указ Президента Республики Узбекистан "О дальнейшем развитии компьютеризации и внедрении информационно-коммуникационных технологий". Данным Указом определены первоочередные задачи в сфере развития и внедрения современных систем компьютеризации и информационно-коммуникационных технологий. К ним относятся:

- широкое внедрение компьютерных и информационных технологий в отраслях реальной экономики, в сфере управления, бизнеса, науки и образования, создание условий для широкого доступа различных слоев населения к современным компьютерным и информационным системам;
- введение в учебный процесс в школах, профессиональных колледжах, академических лицеях и высших учебных заведениях прогрессивных систем обучения, основанных на овладении и активном использовании современных компьютерных и информационных технологий;
- организация подготовки высококвалифицированного кадрового потенциала для работы в области информационно-коммуникационных технологий, в том числе в сфере создания программных средств, информационных баз данных, формирования республиканских, отраслевых и локальных информационно-коммуникационных сетей, разработки компьютерной и телекоммуникационной техники;
- внедрение высокоскоростного доступа к национальным и международным информационным сетям, обеспечение доступа к ним населенных пунктов, включая сельские;
- ускоренное развития технической инфраструктуры информационно-коммуникационных технологий на всей территории страны, включая мобильную связь, IP-технологии, другие современные средства

телекоммуникации и передачи данных, с учетом конвергенции информационно-коммуникационных сетей и услуг.

Данным Указом был образован Координационный Совет по развитию компьютеризации и информационно-коммуникационных технологий при Кабинете Министров Республики, председателем которого назначен заместитель Премьер-министра Республики Узбекистан, генеральный директор Узбекского агентства связи и информатизации.

Узбекское агентство связи и информатизации определено рабочим органом Координационного Совета по развитию компьютеризации и информационно-коммуникационных технологий.

Во исполнение Указа Президента Кабинет Министров Республики Узбекистан утвердил постановлением от 6 июня 2002 года № 200 Программу развития компьютеризации и информационно-коммуникационных технологий на 2002-2010 годы, в которой определены целевые ориентиры развития телекоммуникаций и передачи данных, использования ресурсов, создания собственных сайтов в интернете.

1. C++ - язык возможностей

C++ в настоящее время считается господствующим языком, используемым для разработки коммерческих программных продуктов. В последние годы это господство слегка поколебалось вследствие аналогичных претензий со стороны такого языка программирования, как Java, но маятник общественного мнения качнулся в другую сторону, и многие программисты, которые бросили C++ ради Java, в последнее время поспешили вернуться к своей прежней привязанности. В любом случае эти два языка настолько похожи, что, изучив один из них, вы автоматически осваиваете 90% другого.

C# — это новый язык, разработанный Microsoft для сетевой платформы. По существу C# является разновидностью C++, и несмотря на ряд принципиальных отличий, языки C# и C++ совпадают примерно на 90%. Вероятно, пройдет немало времени, прежде чем язык C# составит серьезную конкуренцию языку C++; но даже если это и произойдет, то знание языка C++ окажется существенным преимуществом.

C++ является языком программирования общего назначения. Естественная для него область применения - системное программирование, понимаемое в широком смысле этого слова. Кроме того, C++ успешно используется во многих областях приложения, далеко выходящих за указанные рамки. Реализации C++ теперь есть на всех машинах, начиная с самых скромных микрокомпьютеров - до самых больших супер-ЭВМ, и практически для всех операционных систем.

Безусловно C++ многим обязан языку C, который сохраняется как его подмножество. Сохранены и все свойственные C средства низкого уровня, предназначенные для решения самых насущных задач системного программирования. C, в свою очередь, многим обязан своему предшественнику языку BCPL . Комментарий языка BCPL был восстановлен в C++. Еще одним источником вдохновения был язык SIMULA-67; именно из него была заимствована концепция классов (вместе с производными классами и виртуальными функциями). Возможность в C++ перегрузки операций и свобода размещения описаний всюду, где может встречаться оператор, напоминают язык Алгол-68.

Название C++ (си плюс плюс) , было придумано РикомМаскитти летом 1983 г. Это название отражает эволюционный характер изменений языка C. Обозначение ++ относится к операции наращивания C. Чуть более короткое имя C+ является синтаксической ошибкой. Кроме того, оно уже было использовано как название совсем другого языка. Знатоки семантики C находят, что C++ хуже, чем ++C. Язык не получил названия D, поскольку он является расширением C, и в нем не делается попыток решить какие-либо проблемы за счет отказа от возможностей C. Еще одну интересную интерпретацию названия C++ можно найти в приложении к.

Изначально C++ был задуман для того, чтобы автору и его друзьям не надо было программировать на ассемблере, C или других современных языках высокого уровня. Основное его предназначение - упростить и сделать более приятным процесс программирования для отдельного программиста. До недавнего времени не было плана разработки C++ на бумаге. Проектирование, реализация и документирование шли параллельно. Никогда не существовало "проекта C++" или "Комитета по разработке C++". Поэтому язык развивался и продолжает развиваться так, чтобы преодолеть все проблемы, с которыми столкнулись пользователи. Толчками к развитию служат также и обсуждения автором всех проблем с его друзьями и коллегами.

2. Сравнение языков C++ и C

Выбор C в качестве базового языка для C++ объясняется следующими его достоинствами:

- универсальность, краткость и относительно низкий уровень;
- адекватность большинству задач системного программирования;
- он идет в любой системе и на любой машине;
- полностью подходит для программной среды UNIX.

В C существуют свои проблемы, но в языке, разрабатываемом "с нуля" они появились бы тоже, а проблемы C, по крайней мере, хорошо известны. Более важно то, что ориентация на C позволила использовать язык "C с классами" как полезный (хотя и не очень удобный) инструмент в течение первых месяцев раздумий о введении в C классов в стиле Симулы.

C++ стал использоваться шире, но по мере роста его возможностей, выходящих за пределы C, вновь и вновь возникала проблема совместимости. Ясно, что отказавшись от части наследства C, можно избежать некоторых проблем. Это не было сделано по следующим причинам:

1. существуют миллионы строк программ на C, которые можно улучшить с помощью C++, но при условии, что полной переписи их на язык C++ не потребуется;

2. существуют миллионы строк библиотечных функций и служебных программ на C, которые можно было бы использовать в C++ при условиях совместимости обоих языков на стадии связывания и их большого синтаксического сходства;

3. существуют сотни тысяч программистов, знающих C; им достаточно овладеть только новыми средствами C++ и не надо изучать основ языка;

4. поскольку C и C++ будут использоваться одними и теми же людьми на одних и тех же системах многие годы, различия между языками должны быть либо минимальными, либо максимальными, чтобы свести к минимуму количество ошибок и недоразумений. Описание C++ было переработано так, чтобы гарантировать, что любая допустимая в обоих языках конструкция означала в них одно и то же.

Как язык, так и стандартные библиотеки C++ проектировались в расчете на переносимость. Имеющиеся реализации языка будут работать в большинстве систем, поддерживающих C. В программах на C++ можно использовать библиотеки C. Большинство служебных программ, рассчитанных на C, можно использовать и в C++.

Язык С сам развивался в последние несколько лет, что отчасти было связано с разработкой С++. Стандарт ANSI для С содержит, например, синтаксис описания функций, позаимствованный из языка "С с классами". Происходит взаимное заимствование, например, тип указателя void* был придуман для ANSI С, а впервые реализован в С++. Теперь С++ более совместим с языком С, чем это было вначале. В идеале С++ должен максимально приближаться к ANSI С, но не более. Стопроцентной совместимости никогда не было и не будет, поскольку это нарушит надежность типов и согласованность использования встроенных и пользовательских типов, а эти свойства всегда были одними из главных для С++.

Для изучения С++ не обязательно знать С. Программирование на С способствует усвоению приемов и даже трюков, которые при программировании на С++ становятся просто ненужными. Например, явное преобразование типа (приведение) , в С++ нужно гораздо реже, чем в С. Тем не менее, хорошие программы на языке С по сути являются программами на С++. Например, все программы из классического описания С являются программами на С++. В процессе изучения С++ будет полезен опыт работы с любым языком со статическими типами.

3. Эффективность и структура

Развитие языка C++ происходило на базе языка C, и, за небольшим исключением, C был сохранен в качестве подмножества C++. Базовый язык C был спроектирован таким образом, что имеется очень тесная связь между типами, операциями, операторами и объектами, с которыми непосредственно работает машина, т.е. числами, символами и адресами. За исключением операций new, delete и throw, а также проверяемого блока, для выполнения операторов и выражений C++ не требуется скрытой динамической аппаратной или программной поддержки.

Первоначально язык C задумывался как конкурент ассемблера, способный вытеснить его из основных и наиболее требовательных к ресурсам задач системного программирования. В проекте C++ были приняты меры, чтобы успехи C в этой области не оказались под угрозой. Различие между двумя языками прежде все состоит в степени внимания, уделяемого типам и структурам. Язык C выразителен и в то же время снисходителен по отношению к типам. Язык C++ еще более выразителен, но такой выразительности можно достичь лишь тогда, когда типам уделяют большое внимание. Когда типы объектов известны, транслятор правильно распознает такие выражения, в которых иначе программисту пришлось бы записывать операции с утомительными подробностями. Кроме того, знание типов позволяет транслятору обнаруживать такие ошибки, которые в противном случае были бы выявлены только при тестировании. Отметим, что само по себе использование строгой типизации языка для контроля параметров функции, защиты данных от незаконного доступа, определения новых типов и операций не влечет дополнительных расходов памяти и увеличения времени выполнения программы.

В проекте C++ особое внимание уделяется структурированию программы. Это вызвано увеличением размеров программ со времени появления C. Небольшую программу (скажем, не более 1000 строк) можно заставить из упрямства работать, нарушая все правила хорошего стиля программирования. Однако, действуя так, человек уже не сможет справиться с большой программой. Если у вашей программы в 10 000 строк плохая структура, то вы обнаружите, что новые ошибки появляются в ней так же быстро, как удаляются старые. C++ создавался с целью, чтобы большую программу можно было структурировать таким образом, чтобы одному человеку не пришлось работать с текстом в 25000 строк. В настоящее время можно считать, что эта цель полностью достигнута.

Существуют, конечно, программы еще большего размера. Однако те из них, которые действительно используются, обычно можно разбить на несколько практически независимых частей, каждая из которых имеет значительно меньший упомянутого размер. Естественно, трудность написания и сопровождения программы определяется не только числом строк текста, но и сложностью предметной области. Так что приведенные

здесь числа, которыми обосновывались наши соображения, не надо воспринимать слишком серьезно.

К сожалению, не всякую часть программы можно хорошо структурировать, сделать независимой от аппаратуры, достаточно понятной и т.д. ВС++ есть средства, непосредственно и эффективно представляющие аппаратные возможности. Их использование позволяет избавиться от беспокойства о надежности и простоте понимания программы. Такие части программы можно скрывать, предоставляя надежный и простой интерфейс с ними.

Естественно, если С++ используется для большой программы, то это означает, что язык используют группы программистов. Полезную роль здесь сыграют свойственные языку модульность, гибкость и строго типизированные интерфейсы. ВС++ есть такой же хороший набор средств для создания больших программ, как во многих языках. Но когда программа становится еще больше, проблемы по ее созданию и сопровождению перемещаются из области языка в более глобальную область программных средств и управления проектом.

Основное внимание уделяется методам создания универсальных средств, полезных типов, библиотек и т.д. Эти методы можно успешно применять как для маленьких, так и для больших программ. Более того, поскольку все нетривиальные программы состоят из нескольких в значительной степени независимых друг от друга частей, методы программирования отдельных частей пригодятся как системным, так и прикладным программистам.

Может возникнуть подозрение, что запись программы с использованием подробной системы типов, увеличит размер текста. Для программы на С++ это не так: программа на С++, в которой описаны типы формальных параметров функций, определены классы и т.п., обычно бывает даже короче своего эквивалента на С, где эти средства не используются. Когда в программе на С++ используются библиотеки, она также оказывается короче своего эквивалента на С, еслион существует.

4. КРАТКИЙ ОБЗОР C++

Язык программирования C++ задумывался как язык, который будет:

- лучше языка C;

- поддерживать абстракцию данных;

-поддерживать объектно-ориентированное программирование.

C++ - язык общего назначения и задуман для того, чтобы настоящие программисты получили удовольствие от самого процесса программирования. За исключением второстепенных деталей он содержит язык C как подмножество. Язык C расширяется введением гибких и эффективных средств, предназначенных для построения новых типов. Программист структурирует свою задачу, определив новые типы, которые точно соответствуют понятиям предметной области задачи. Такой метод построения программы обычно называют абстракцией данных. Информация о типах содержится в некоторых объектах типов, определенных пользователем. С такими объектами можно работать надежно и просто даже в тех случаях, когда их тип нельзя установить на стадии трансляции. Программирование с использованием таких объектов обычно называют объектно-ориентированным. Если этот метод применяется правильно, то программы становятся короче и понятнее, а сопровождение их упрощается.

Ключевым понятием C++ является класс. Класс - это определяемый пользователем тип. Классы обеспечивают упрятывание данных, их инициализацию, неявное преобразование пользовательских типов, динамическое задание типов, контролируемое пользователем управление памятью и средства для перегрузки операций. В языке C++ концепции контроля типов и модульного построения программ реализованы более полно, чем в C. Кроме того, C++ содержит усовершенствования, прямо с классами не связанные: символические константы, функции-подстановки, стандартные значения параметров функций, перегрузка имен функций, операции управления свободной памятью и ссылочный тип. В C++ сохранены все возможности C эффективной работы с основными объектами, отражающими аппаратную "реальность" (разряды, байты, слова, адреса и т.д.). Это позволяет достаточно эффективно реализовывать пользовательские типы.

Объектно-ориентированное программирование - это метод программирования, способ написания "хороших" программ для множества задач. Если этот термин имеет какой-то смысл, то он должен подразумевать: такой язык программирования, который предоставляет хорошие возможности для объектно-ориентированного стиля программирования.

Нельзя сказать, что один язык лучше другого только потому, что в нем есть возможности, которые в другом отсутствуют. Часто бывает как раз наоборот. Здесь более важно не то, какими возможностями обладает язык, а то, насколько имеющиеся в нем возможности поддерживают избранный стиль программирования для определенного круга задач.

Язык C++ проектировался для поддержки абстракции данных и объектно-ориентированного программирования в добавление к традиционному стилю C. Впрочем, это не значит, что язык требует какого-то одного стиля

5. Переменные и операции языка С++

Буквы и цифры

Множество символов Си включает большие и малые буквы из английского алфавита и 10 десятичных арабских цифр:

-большие английские буквы: A B C D E F G H I J K L M N O P Q R T U V W X Y Z

-малые английские буквы: a b c d e f g h i j k l m n o p q r t u v w x y z

-десятичные цифры: 0 1 2 3 4 5 6 7 8 9

Буквы и цифры используются при формировании констант, идентификаторов и ключевых слов. Все эти конструкции описаны ниже. Компилятор Си рассматривает одну и ту же малую и большую буквы как отличные символы. Если в данной записи использованы малые буквы, то замена малой буквы "a" на большую букву "A" сделает отличной данную запись от предшествующей.

Пробельные символы

Пробел, табуляция, перевод строки, возврат каретки, новая страница, вертикальная табуляция и новая строка- это сиволы, называемые пробельными, поскольку они имеют то же самое назначение, как и пробелы между словами и строками на печатной странице. Эти символы разделяют объекты, определенные пользователем, такие, как константы и идентификаторы, от других объектов программы.

Символ CONTROL-Z рассматривается как индикатор конца файла. Компилятор игнорирует любой текст, следующий за символом

CONTROL-Z.

Компилятор Си игнорирует пробельные символы, если они не используются как разделители или как компоненты константы-символа или строковых литералов. Это нужно иметь в виду, чтобы дополнительно использовать пробельные символы для повышения наглядности программы (например,для просмотра редактором текстов).

Знаки пунктуации и специальные символы

Эти символы имеют специальный смысл для компилятора Си. Их использование в языке Си описывается в дальнейшем содержании руководства. Знаки пунктуации из множества представимых символов, которые не представлены в данном списке, могут быть использованы только в строковых литералах, константах-символах и комментариях.

ESC- последовательности

ESC- последовательности- это специальные символьные комбинации, которые представляют пробельные символы и неграфические символы в строках и символьных константах.

Их типичное использование связано со спецификацией таких действий, как возврат каретки и табуляция , а также для задания литеральных

представлений символов, таких как символ двойная кавычка. ESC-последовательность состоит из наклонной черты влево, за которой следует буква, знаки пунктуации ' " \ или комбинация цифр.

Если наклонная черта влево предшествует символу, не включенному в этот список, то наклонная черта влево игнорируется, а символ представляется как литеральный. Например, изображение \c представляет символ "c" в литеральной строке или константе-символе.

Последовательности \ddd и \xdd позволяют задать любой символ в ASCII (Американский стандартный код информационного интерфейса) как последовательность трех восьмеричных цифр или двух шестнадцатеричных цифр. Например, символ пробела может быть задан как \010 или \x08. Код ASCII "нуль" может быть задан как \0 или \x0 . В восьмеричной ESC-последовательности могут быть использованы от одной до трех восьмеричных цифр.

Например, символ пробела может быть задан как \10 . Точно так же в шестнадцатеричной ESC- последовательности могут быть использованы от одной до двух шестнадцатеричных цифр. Так, шестнадцатеричная последовательность для символа пробела может быть задана как \x08 или \x8

Операции

Операции- это специальные комбинации символов, специфицирующие действия по пробразованию различных величин. Компилятор интерпретирует каждую из этих комбинаций как самостоятельную единицу, называемую лексемой (token).

Операция условного выражения ?: -это тернарная, а не двухсимвольная операция. Формат условного выражения следующий: <expression>?<expression>:<expression>

Константы

Константа- это число, символ или строка символов. Константы используются в программе как неизменяемые величины. В языке Си различают четыре типа констант: целые константы, константы с плавающей точкой, константы-символы и строчные литералы.

Целые константы

Целая константа- это десятичное, восьмеричное или шестнадцатеричное число, которое представляет целую величину. Десятичная константа имеет следующий формат представления:

<digits>, где <digits> - это одна или более десятичных цифр от 0 до 9.

Восьмеричная константа имеет следующий формат представления:

0<odigits>,

где <odigits> - это одна или более восьмеричных цифр от 0 до 7. Запись ведущего нуля необходима.

Шестнадцатеричная константа имеет один из следующих форматов представления:

0x<hdigits>

0X<hdigits>,

где <hdigits> одна или более шестнадцатеричных цифр. Шестнадцатеричная цифра может быть цифрой от 0 до 9 или

буквой (большой или малой) от A до F. В представлении константы допускается "смесь" больших и малых букв. Запись ведущего нуля и следующего за ним символа x или X необходима.

Пробельные символы не допускаются между цифрами целой константы.

Целые константы всегда специфицируют положительные величины. Если требуется отрицательные величины, то необходимо сформировать константное выражение из знака минус и следующей за ним константы. Знак минус рассматривается как арифметическая операция.

Каждая целая константа специфицируется типом, определяющим ее представление в памяти и область значений. Десятичные константы могут быть типа int или long.

Восьмеричные и шестнадцатеричные константы в зависимости от размера могут быть типа int, unsignedint, long или unsignedlong. Если константа может быть представлена как int, она специфицируется типом int. Если ее величина больше, чем максимальная положительная величина, которая может быть представлена типом int, но меньше величины, которая представляется в том же самом числе бит как и int, она задается типом unsignedint. Наконец, константа, величина которой больше чем максимальная величина, представляемая типом unsignedint, задется типом long или unsignedlong, если это необходимо.

Важность рассмотренных выше правил состоит в том, что восьмеричные и шестнадцатеричные константы не содержат "знаковых" расширений, когда они преобразуются к более длинным типам.

Программист может определить для любой целой константы тип long, приписав букву "l" или "L" в конец константы. В Табл. 2.6 показаны примеры целых констант.

Константы с плавающей точкой

Константа с плавающей точкой- это действительное десятичное положительное число. Величина действительного числа включает целую, дробную части и экспоненту. Константы с плавающей точкой имеют следующий формат представления:

[<digits>][.<digits>][E[-]<digits>],

где <digits> - одна или более десятичных цифр (от 0 до 9), а E или e - символ экспоненты. Целая или дробная части константы могут быть опушены, но не обе сразу. Десятичная точка может быть опущена только тогда, когда задана экспонента.

Экспонента состоит из символа экспоненты, за которым следует целочисленная величина экспоненты, возможно отрицательная.

Пробельные символы не могут разделять цифры или символы константы.

Константы с плавающей точкой всегда специфицируют положительные величины. Если требуются отрицательные величины, то необходимо сформировать константное выражение из знака минус и следующей за ним константы. Знак минус рассматривается как арифметическая операция.

Примеры констант с плавающей точкой и константных выражений:

15.75
1.575E1
1575e-2
-0.0025
-2.5e-3
25e-4

Целая часть константы с плавающей точкой может быть опущена, например:

.75
.0075e2
-.125
-.175E-2

Все константы с плавающей точкой имеют тип double.

Константа-символ

Константа-символ- это буква, цифра, знак пунктуации или ESC- символ, заключенные в одиночные кавычки. Величина константы-символа равна значению представляющего ее кода символа.

Константа-символ имеет следующую форму представления:

'<char>',

где <char> может быть любым символом иэ множества представимых символов, включая любой ESC- символ, исключая одиночную кавычку ('), наклонную черту влево (\) и символ новой строки.

Чтобы использовать одиночную кавычку или наклонную черту влево в качестве константы-символа, необходимо вставить перед этими знаками наклонную черту влево. Чтобы представить символ новой строки, необходимо использовать запись '\n'.

Идентификаторы

Идентификаторы- это имена переменных, функций и меток, используемых в программе. Идентификатор создается объявлением соответствующей ему переменной или функции.После этого его можно использовать в последующих операторах программы. Идентификатор- это последовательность из одной или более букв, цифр или подчерков(_), которая начинается с буквы или подчерка. Допускается любое число символов в идентификаторе, однако только первые 31 символ распознаются компилятором. (Программы, использующие результат работы компилятора, такие как, линкер, могут распознавать меньшее число символов).

При использовании подчерков в идентификаторе нужно быть осторожным, поскольку идентификаторы, начинающиеся с подчерка могут совпадать (войти в конфликт) с именами "скрытых" системных программ.

Примеры идентификаторов:

temp1

toofpage

skip12

Компилятор Си рассматривает буквы верхнего и нижнего регистров как различные символы. Поэтому можно создать отдельные независимые идентификаторы, которые совпадают орфографически, но различаются большими и малыми буквами. Например, каждый из следующих идентификаторов является уникальным:

add

ADD

Add

aDD

Компилятор Си не допускает идентификаторов, которые имеют ту же самую орфографию, что и ключевые слова.

Ключевые слова

Ключевые слова- это предопределенные идентификаторы, которые имеют специальное значение для компилятора Си. Их можно использовать только так как они определены. Имена объектов программы не могут совпадать с названиями ключевых слов.

Списокключевыхслов:

auto double intstruct

break else long switch

caseenum register typedef

char extern return union

const float short unsigned

continue for signed void

defaultgotosizeof while

do if static volatile

Ключевые слова не могут быть переопределены. Тем не менее, они могут быть названы другим текстом, но тогда перед компиляцией они должны быть заменены посредством препроцессора на соответствующие ключевые слова.

Ключевые слова const и volatile зарезервированы для будущего использования.

Комментарии

Комментарий- это последовательность символов, которая воспринимается компилятором как отдельный пробельный символ или, другими словами, игнорируется.

Комментарий имеет следующую форму представления:

/*<characters>*/,

где <characters> может быть любой комбинацией символов из множества представимых символов, включая символы новой строки, но исключая комбинацию */. Это означает, что комментарии могут занимать более одной строки, но не могут быть вложенными.

Комментарии допускаются везде, где разрешены пробельные символы. Компилятор игнорирует символы комментария, в частности, в комментариях допускается запись ключевых слов и зто не приведет к ошибке. Так как компилятор рассматривает комментарий как символ пробела, то комментарии не могут появляться внутри лексем.

Следующие примеры иллюстрируют некоторые комментарии:

```
/* Comments can separate and document
lines of a program. */
 /* Comments can contain keywords such as for
and while */
/*****************************************
 Comments can occupy several lines.
*****************************************/
```

Так как комментарии не могут содержать вложенных комментариев, то следующий пример будет ошибочным:

```
/* Youcannot/* nest */ comments */
```

Компилятор распознает первую комбинацию */ после слова nest как конец комментария. Затем, компилятор попытается обрабатывать оставшийся текст и выработает сообщение об ошибке. Чтобы обойти компиляцию комментариев больших размеров, нужно использовать директиву #if препроцессора.

Лексемы

Когда компилятор обрабатывает программу, он разбивает программу на группы символов, называемых лексемами. Лексема- это единица текста программы, которая имеет определенный смысл для компилятора и которая не может быть разбита в дальнейшем. Операции, константы, идентификаторы и ключевые слова, описанные в этом разделе,являются примерами лексем. Знаки пунктуации, такие как квадратные скобки ([]), фигурные скобки ({}), угловые скобки (<>), круглые скобки и запятые, также являются лексемами. Границы лексем определяются пробельными символами и другими лексемами, такими как операции и знаки пунктуации. Чтобы предупредить неправильную работу компилятора, запрещаются пробельные символы между символами

идентификаторов, операциями, состоящими из нескольких символов и символами ключевых слов.

Когда компилятор выделяет отдельную лексему, он последовательно объединяет столько символов, сколько возможно, прежде чем перейти к обработке следующей лексемы. Поэтому лексемы, не разделенные пробельными символами, могут быть проинтерпретированы неверно.

Например, рассмотрим следующее выражение:

i+++j

В этом примере компилятор вначале создает из трех знаков плюс самую длинную из возможных операций (++), а затем обработает оставшийся знак +, как операцию сложения (+). Выражение проинтерпретируется как (i++)+(j), а не как (i)+(++j). В таких случаях необходимо использовать пробельные символы или круглые скобки, чтобы однозначно определить ситуацию.

Заключение

C++ — компилируемый, статически типизированный язык программирования общего назначения.

Поддерживает такие парадигмы программирования, как процедурное программирование, объектно-ориентированное программирование, обобщённое программирование, обеспечивает модульность, раздельную компиляцию, обработку исключений, абстракцию данных, объявление типов (классов) объектов, виртуальные функции. Стандартная библиотека включает, в том числе, общеупотребительные контейнеры и алгоритмы. C++ сочетает свойства как высокоуровневых, так и низкоуровневых языков. В сравнении с его предшественником — языком C, — наибольшее внимание уделено поддержке объектно-ориентированного и обобщённого программирования.

C++ широко используется для разработки программного обеспечения, являясь одним из самых популярных языков программирования. Область его применения включает создание операционных систем, разнообразных прикладных программ, драйверов устройств, приложений для встраиваемых систем, высокопроизводительных серверов, а также развлекательных приложений (игр). Существует множество реализаций языка C++, как бесплатных, так и коммерческих и для различных платформ. Например, наплатформе x86 это GCC, Visual C++, Intel C++ Compiler, Embarcadero (Borland) C++ Builder идругие. C++ оказал огромное влияние на другие языки программирования, в первую очередь на Java и C#.

Синтаксис C++ унаследован от языка C. Одним из принципов разработки было сохранение совместимости с C. Тем не менее, C++ не является в строгом смысле надмножеством C; множество программ, которые могут одинаково успешно транслироваться как компиляторами C, так и компиляторами C++, довольно велико, но не включает все возможные программы на C

Язык C++ явился мощным и стремительным рывком в развитии программирования. C++ и по сей день занимает господствующее положение среди языков программирования в мире. Огромное множество профессиональных программистов использует именно его при разработке разного рода проектов. Очевидно, этот язык будет сохранять свое солидное положение ещё не один год, при этом по-прежнему развиваясь и совершенствуясь.

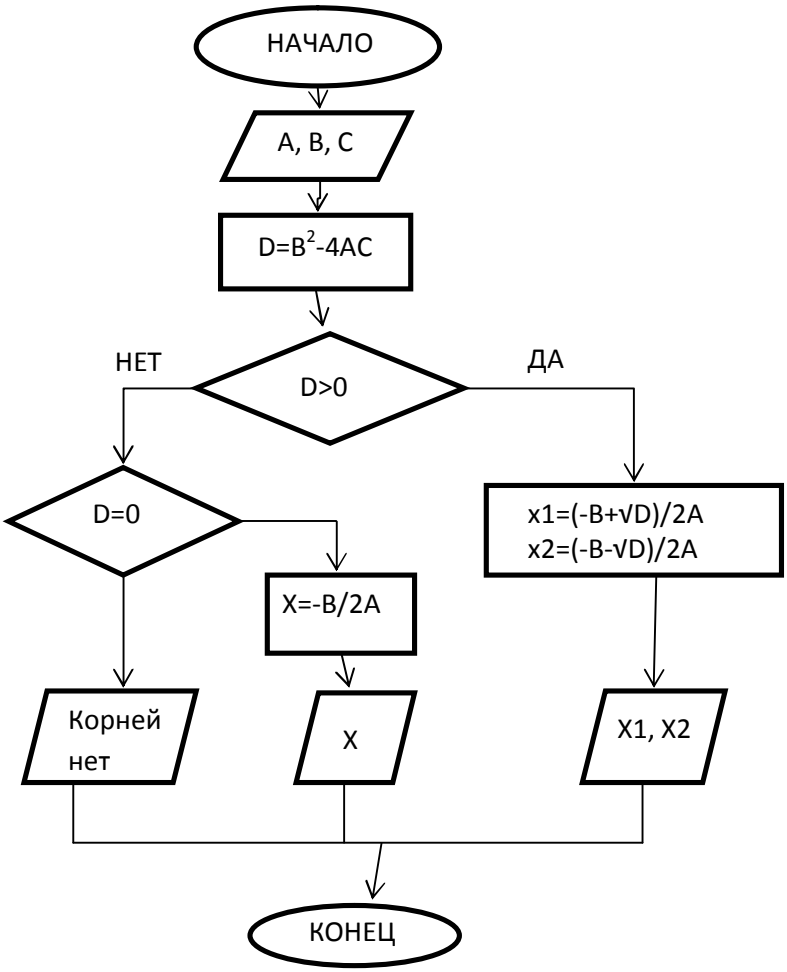
Список литературы

1. «Бьерн Страуструп. Язык программирования C++»
2. *Джесс либерти*, «Освой самостоятельно C++ за 21 день», изд. Дом «Вильямс», Москва - Санкт-Петербург – Киев, 2001
3. *Н. Секунов*, «Самоучитель VisualC++ 6», изд. «БХВ-Петербург», Санкт-Петербург, 2003

Приложения

Задание 1. Составить алгоритм и программу на С++ для нахождения корней квадратного уравнения, путем введения его коэффициентов.

Решение:




```
#include <iostream>
#include "math.h"
using namespace std;
int main()
{int a,b,c,x1,x2,d;

cout<<"Vvedite koefficient a"<<endl;
cin>>a;
cout<<"Vvedite koefficient b"<<endl;
cin>>b;
cout<<"Vvedite koefficient c"<<endl;
cin>>c;
```

```
   d=pow(b,2)-4*a*c;
if(d==0) { x1=-b/(2*a); cout<<"\n\n  x1="<<x1<<endl; }
if(d>0)
    {
x1=(-b-sqrt(d))/(2*a); cout<<"\n\n  x1="<<x1<<endl;
x2=(-b+sqrt(d))/(2*a); cout<<"  x2="<<x2<<endl;
    }
if(d<0){cout<<"Deystvitelnih korney net" <<endl;
}
return 0;
}
```

Задание 2. Составить алгоритм и программу на С++ для сортировки массива по возрастанию

Решение

```
#include <iostream>
#include "math.h"
using namespace std;
int main()
{intn,i,y[n],j, temp;
cout<<"     Propgamma dlya sortirovki po vozrastaniyu elementov massiva"
<<endl;
cout<<" Vvedite kolichestvo elementov" <<endl;
cout<<"\n N="; cin>>n;
for (i=0; i<n; i++)
{
cout<<"\n Y["<<i<<"]=";
cin>>y[i];
}

for ( i = 0; i< n - 1; ++i)
for ( j = 0; j < n - 1; ++j)
if(y[j] > y[j+1])
        {
temp = y[j];
y[j] = y[j+1];
y[j+1] = temp;
        }
```

```
for ( i = 0; i< n; ++i)
std::cout<< y[i] << " ";
return 0;
}
```