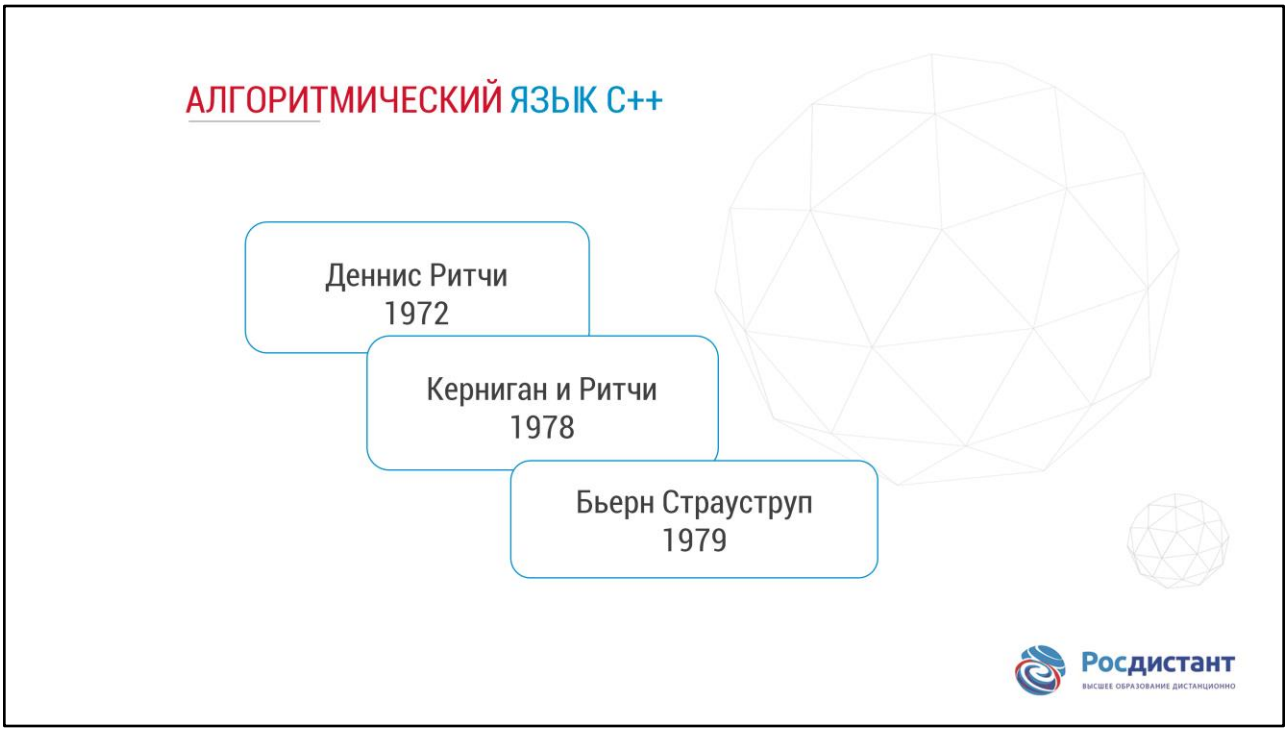

Росдистант
ВЫСШЕЕ ОБРАЗОВАНИЕ ДИСТАНЦИОННО
ОСНОВЫ
ПРОГРАММИРОВАНИЯ
1 ЧАСТЬ



Слайд 2

Тема 1. Базовые средства языка С++

Представляем Вам электронный учебник по курсу Основы программирования. Курс разработан для проекта Росдистант.

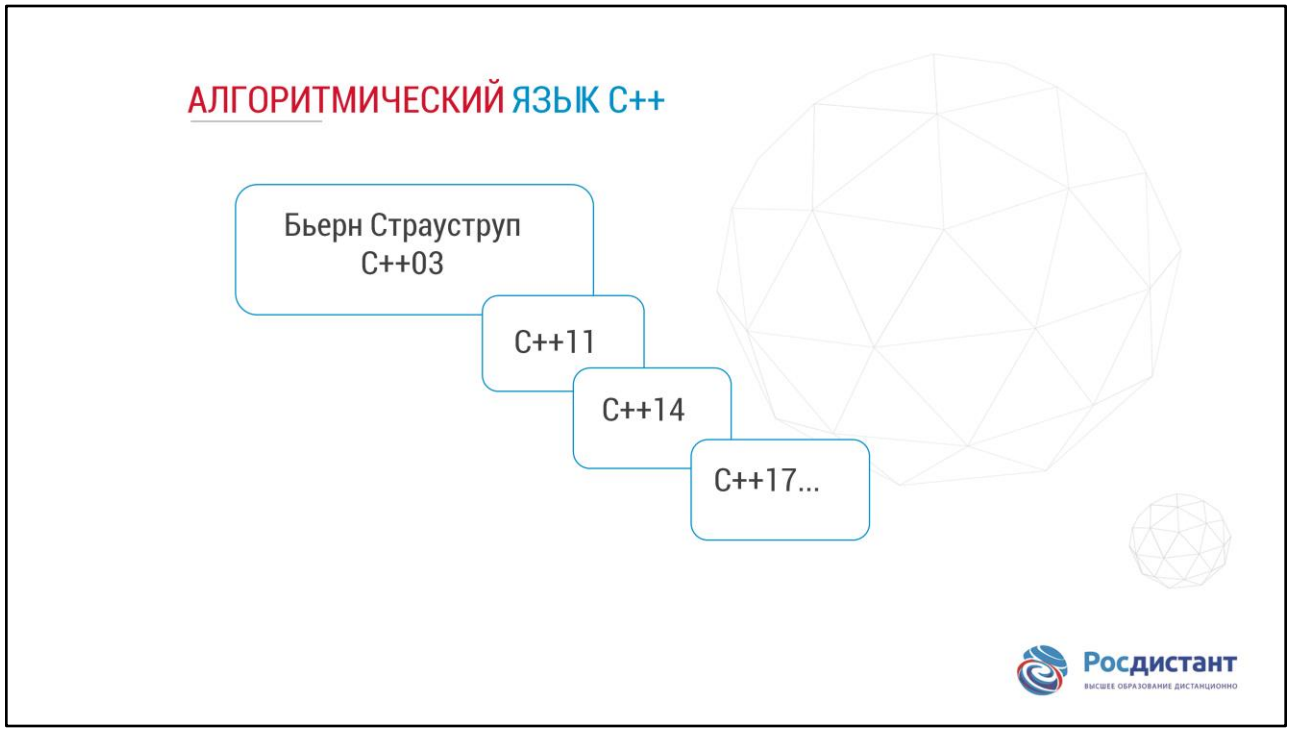
В рамках этого курса мы будем изучать программирование на алгоритмическом языке **С++ .**

«Язык программирования С был разработан в 1972 году Деннисом Ритчи как системный язык программирования, то есть язык для написания операционных систем. Основной задачей Ритчи было создание легко компилируемого минималистического языка. Этот язык предоставлял бы эффективный доступ к памяти, относительно быстро выполнялся, и на котором можно было бы писать эффективный код. Таким образом, при разработке высокоуровневого языка, был создан язык **С ,** который во многом относился к языкам низкого уровня, оставаясь при этом независимым от платформ, для которых мог быть написан код.

Язык **С** имеет ряд существенных особенностей, которые выделяют его среди других языков программирования. В значительной степени на формирование идеологии языка повлияла цель, которую ставили перед собой его создатели – обеспечение системного программиста удобным инструментальным языком, который мог бы заменить Ассемблер.

C в конечном итоге стал настолько эффективным и гибким, что в 1973 году Деннис Ритчи и Кен Томпсон переписали больше половины операционной системы **UNIX ,** используя этот язык. Многие предыдущие операционные системы были написаны на языке ассемблера. Язык **C** и операционная система **UNIX** тесно связаны между собой, и популярность первого отчасти связана с успехом второго.

В 1978 году Брайан Керниган и Деннис Ритчи опубликовали книгу - Язык программирования **C** . Эта книга стала стандартом и своеобразной инструкцией к **C** . Когда требовалась максимальная портативность, то программисты придерживались рекомендаций из этой книги, поскольку большинство компиляторов в то время были реализованы в соответствии со стандартами, присутствующими в этой книге.»



Слайд 3

Язык программирование **С ++** был разработан Бьёрном Страуструпом в качестве дополнения к языку **С** в 1979 г. Он добавил множество новых особенностей в язык С , который был одобрен международной организацией по стандартизации в 1998 году и потом снова в 2003 году, под названием **С ++03 .** Потом были еще три обновления, одобренные в 2011, 2014 и 2017 годах, соответственно, которые добавили больше функциональных возможностей.

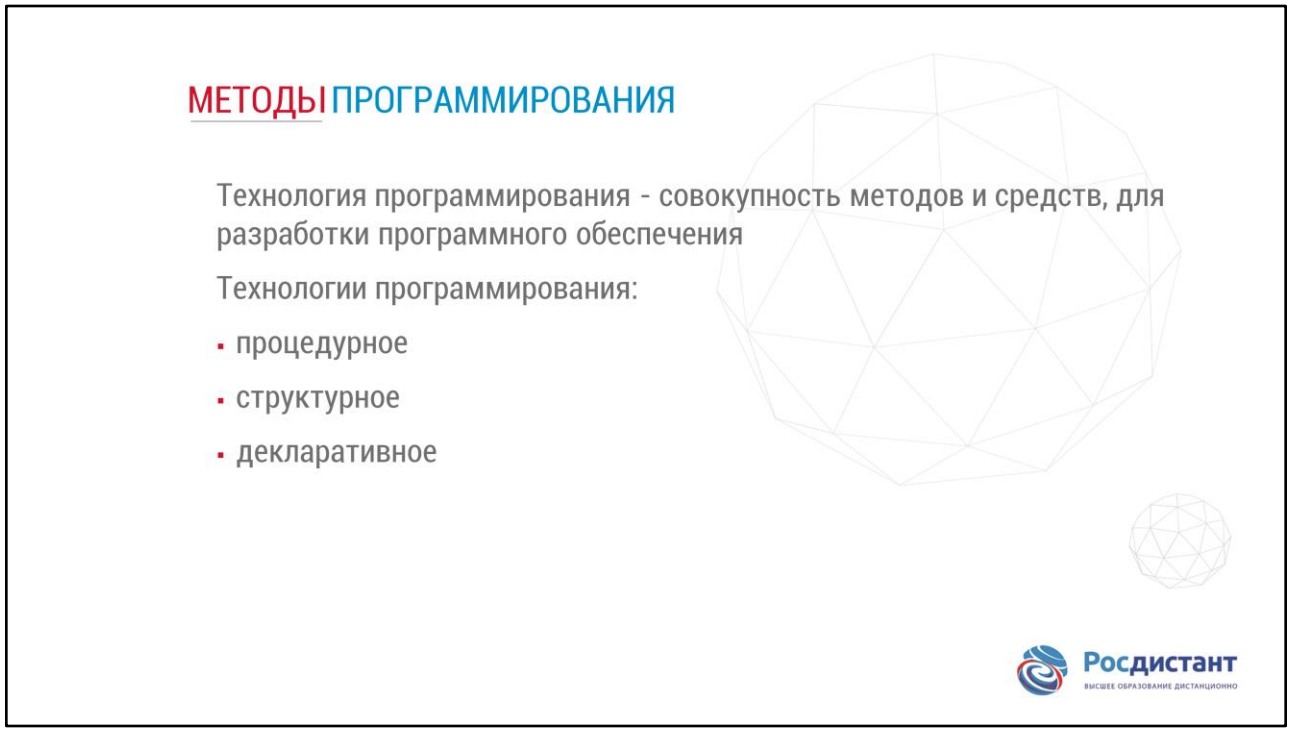
Язык программирование **С ++** был разработан на основе языка **С,** является расширением этого языка, поэтому программы написанные на языке **С** , могут обрабатываться компилятором **С ++.**

«Смысл философии языков **С** , и **С ++** можно определить выражением - доверять программисту. Например, компилятор не будет вам мешать сделать что-то новое, что имеет смысл, но также не будет мешать вам сделать что-то такое, что может привести к сбою. Это одна из главных причин, почему так важно знать то, что вы не должны делать, как и то, что вы должны делать, создавая программы на языках **С** , и **С ++ .**

Недавно был собран комитет по стандартам языка, чтобы завершить работу над последней версией языкового стандарта, **С ++** 20 и определить новую функциональность, которая появится в следующем крупном релизе.

После **С ++** 17 это будет шестая редакция стандарта. Этот язык прошёл долгий

путь от своего бытия над множеством **C .** Программирование на **C ++** включает в себя множество стилей: от простейшего **C** с классами, до написания кода, который выглядит как произведение искусства.»



Слайд 4

Перед тем как приступить к изучению программирования на языке **C++,** рассмотрим технологии программирования и методы программирования. На слайде приведено определение технологии программирования.

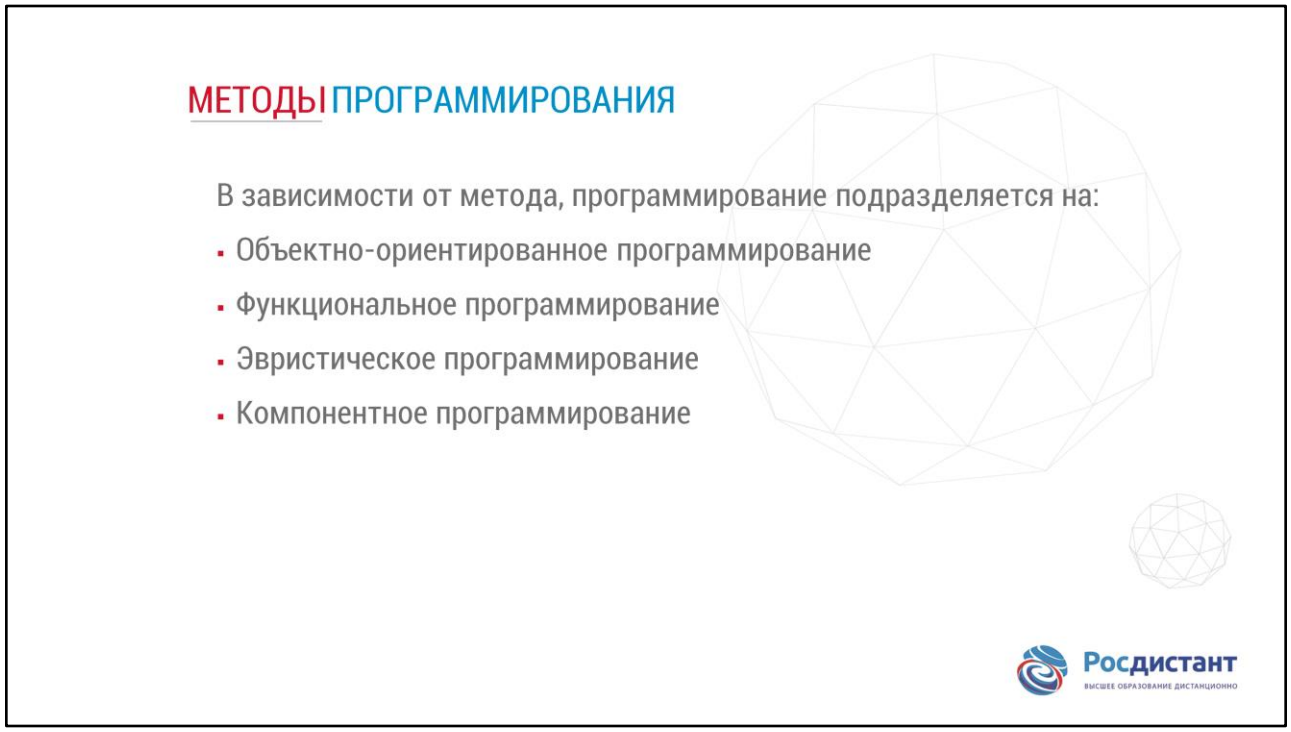
«В технологии должны быть определены последовательность выполнения операций, условия, при которых выполняется каждая операция, описание самих операций: исходные данные, нормативные документы, в том числе стандарты, критерии и методы оценки, результаты и др.

В зависимости от метода, программирование подразделяют на:

- процедурное - метод, в соответствии с которым программы пишутся как перечни последовательно выполняемых команд. При этом используются процедурно-ориентированные языки программирования, например, **PL** , Паскаль, **C** ;
- структурное, модульное - метод написания программ небольшими независимыми структурированными частями, модулями, каждый из которых связан с какой-либо функцией. Основу структурного метода программирования составляет декомпозиция создаваемой системы на отдельные функции и задачи, которые в свою очередь разбиваться на более мелкие. Процесс декомпозиции продолжается вплоть до определения конкретных процедур. Результирующая программа организуется в виде

совокупности взаимосвязанных по определенным правилам модулей. Это упрощает разработку сложных программных продуктов и их тестирование. Языки для модульного программирования – **TurboPascal , C ++** , Ада;

- декларативное - метод, предназначенный для решения задач искусственного интеллекта. В указанном контексте программа описывает логическую структуру решения задачи, указывая преимущественно, что нужно сделать, не вдаваясь в детали. Используются языки программирования типа Пролог.



Слайд 5

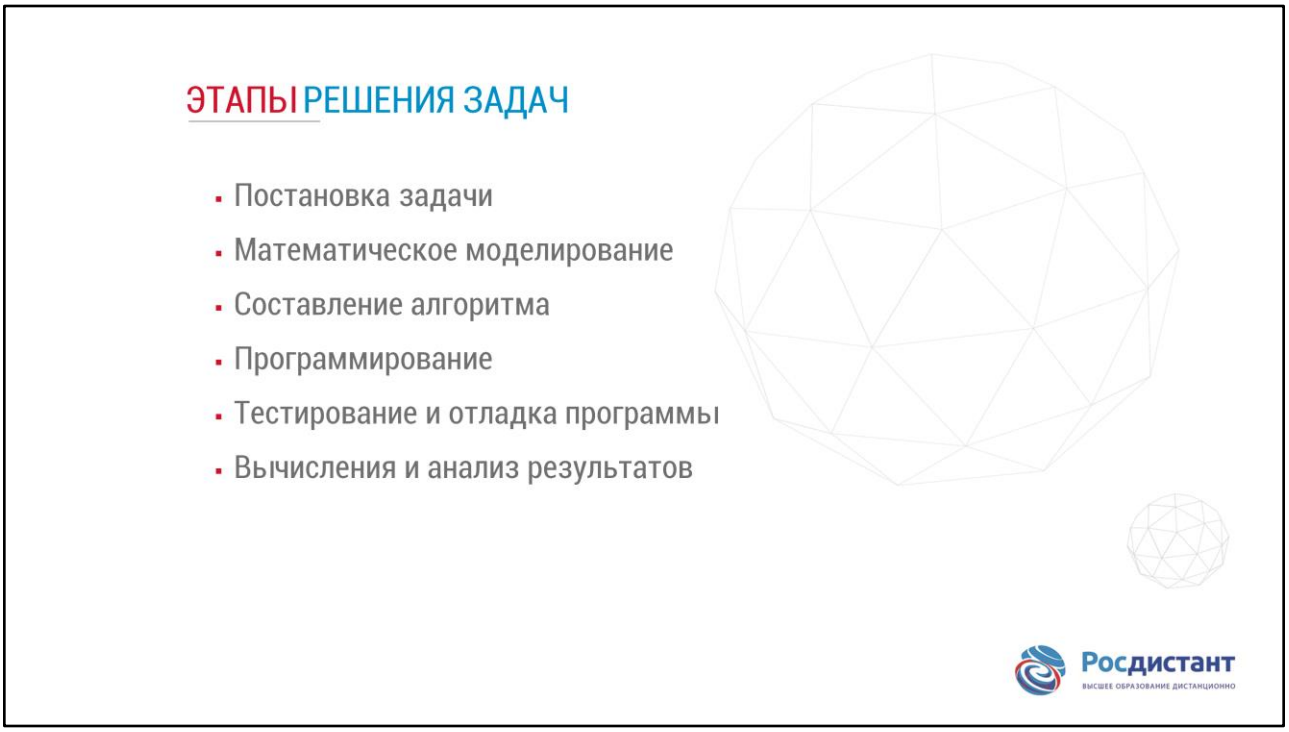
«Объектно-ориентированное программирование представляет собой стратегию проектирования программных средств из объектов. Объект – это нечто, способное пребывать в различных состояниях и имеющее множество операций. Состояние определяется как набор атрибутов объекта. Операции, связанные с объектом, предоставляют сервисы другим объектам для выполнения определенных вычислений. Объекты объединяются в классы, каждый из которых имеет описания всех атрибутов и операций. Объектно-ориентированное программирование – это метод, основанный на использовании концепции объекта, абстрагирующего конкретные его реализации в предметной области. При этом данные тесно связываются с выполняемыми над объектами процедурами.

Функциональное программирование – это метод, основанный на разбиении алгоритма решения задачи на отдельные функциональные модули, а также описании их связей и характера взаимодействия.

Эвристическое программирование – это метод, основанный на моделировании мыслительной деятельности человека. Используется для решения задач, не имеющих строго формализованного алгоритма или связанных с неполнотой исходных данных;

Компонентное программирование – это метод создания программного

обеспечения путем сбора объектов-компонентов в библиотеки или исполняемые файлы. Объекты рассматриваются на логическом уровне проектирования программной системы, а компоненты – как непосредственная физическая реализация объектов. Один компонент может быть реализацией нескольких объектов или даже некоторой части объектной системы, полученной на уровне проектирования. Компоненты в отличие от объектов могут изменяться и пополняться новыми функциями и интерфейсами. Они конструируются в виде некоторой программной абстракции, состоящей из трех частей: информационной, внешней и внутренней.»



Слайд 6

Решение задачи на компьютере проходит указанные на слайде этапы.

«Любая задача начинается со словесного описания, которое называется условием задачи, а в программировании — спецификацией задачи. Далее условие следует формализовать, то есть так записать спецификацию, чтобы можно было ее решить на компьютере. Формализацией условия начинается этап, который называется математической формулировкой задачи. Выводятся формулы и выбираются методы решения задачи, т. е. строится математическая модель задачи. На этом этапе вырабатывается точная формулировка цели задачи, определяется роль компьютера в ее реализации и целесообразность применения компьютера вообще. От корректности постановки задачи пользователем в значительной мере зависят все последующие этапы и успех решения задачи в целом.

Далее в виде алгоритма строится последовательность стандартных действий, выполнение которых даст искомый результат. На выбранном языке программирования проектируется и пишется программа для компьютера. Затем выполняется ее тестирование и отладка на компьютере. Выполняются необходимые расчеты. На слайде приведены этапы решения задач на компьютере.

Осуществляется формализация описания задачи, то есть с помощью

математических, статистических и других методов соотношения между величинами выражаются с помощью математических, логических формул. Вообще, представление объекта исследования, задачи в виде совокупности математических отношений называется его математической моделью.

Приступая к построению модели, обычно вначале анализируют имеющийся опыт решения сходных задач, выясняют, какие методы формализации больше всего подходят для решения поставленной задачи. Метод решения поставленной задачи будет вытекать из принятой математической модели. Если задача может быть решена различными методами, выбирается тот, который наилучшим образом удовлетворяет ее основным требованиям.»

Слайд 7

«Компьютеры понимают только очень ограниченный набор инструкций, и чтобы заставить их что-то делать, нужно четко сформулировать задание, используя эти же инструкции. Программа — это набор инструкций, которые указывают компьютеру, что ему нужно делать. Физическая часть компьютера, которая выполняет эти инструкции, называется – железом, или аппаратной частью.

Процессор компьютера не способен понимать напрямую языки программирования. Очень ограниченный набор инструкций, которые изначально понимает процессор, называется машинным кодом, или машинным языком.

Для решения проблем читабельности кода и чрезмерной сложности были разработаны высокоуровневые языки программирования. Они позволяют писать и выполнять программы, не переживая о совместимости кода с разными архитектурами процессоров.»

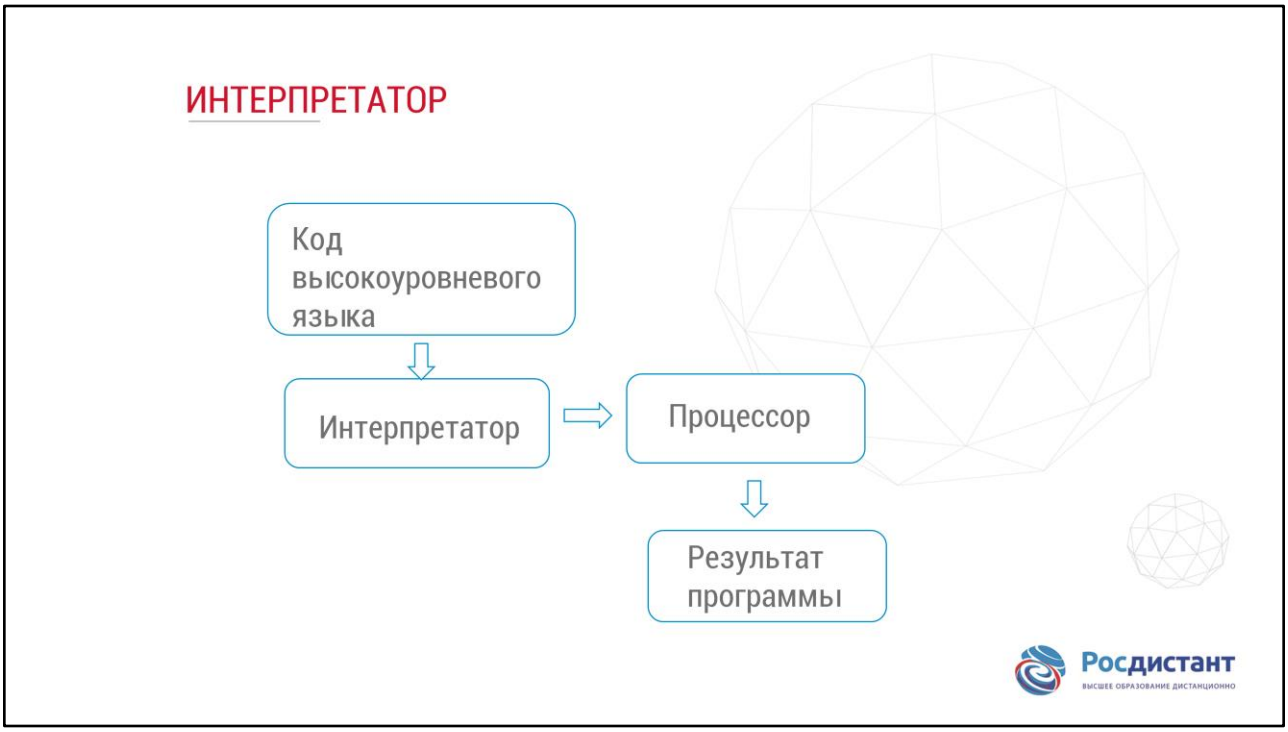
Программы, написанные на языках высокого уровня, также должны быть переведены в машинный код перед выполнением.

Процесс перевода программного кода с алгоритмического языка на язык машинных команд, называется трансляция.

Существует два способа трансляции: компиляция и интерпретация.

Термины компилятор и интерпретатор описывают способ, с помощью которого выполняется программа. Компилятор - это программа, которая читает код и создаёт автономную, способную работать независимо от другого аппаратного или программного обеспечения, исполняемую программу Такую программу процессор понимает напрямую. При запуске программы весь код компилируется целиком, а затем создаётся исполняемый файл и уже при повторном запуске программы компиляция не выполняется.

Процесс компиляции показан на слайде.



Слайд 8

«Линкинг — это процесс связывания всех объектных файлов, генерируемых компилятором, в единую исполняемую программу, которую вы затем сможете запустить, выполнить. Это делается с помощью программы, которая называется линкер, компоновщик.

Кроме объектных файлов, линкер также подключает файлы из Стандартной библиотеки **C ++ ,** или любой другой библиотеки которую вы используете, например, библиотеки графики или звука. Сам по себе язык **C ++** довольно маленький и простой. Тем не менее, к нему подключается большая библиотека дополнительных функций, которые могут использовать ваши программы, и эти функции находятся в Стандартной библиотеке **C ++ .**

После того, как компоновщик закончит линкинг всех объектных файлов при условии, что не будет ошибок, вы получите исполняемый файл.»

Интерпретатор - это программа, которая напрямую выполняет код, без его предыдущей компиляции в исполняемый файл.

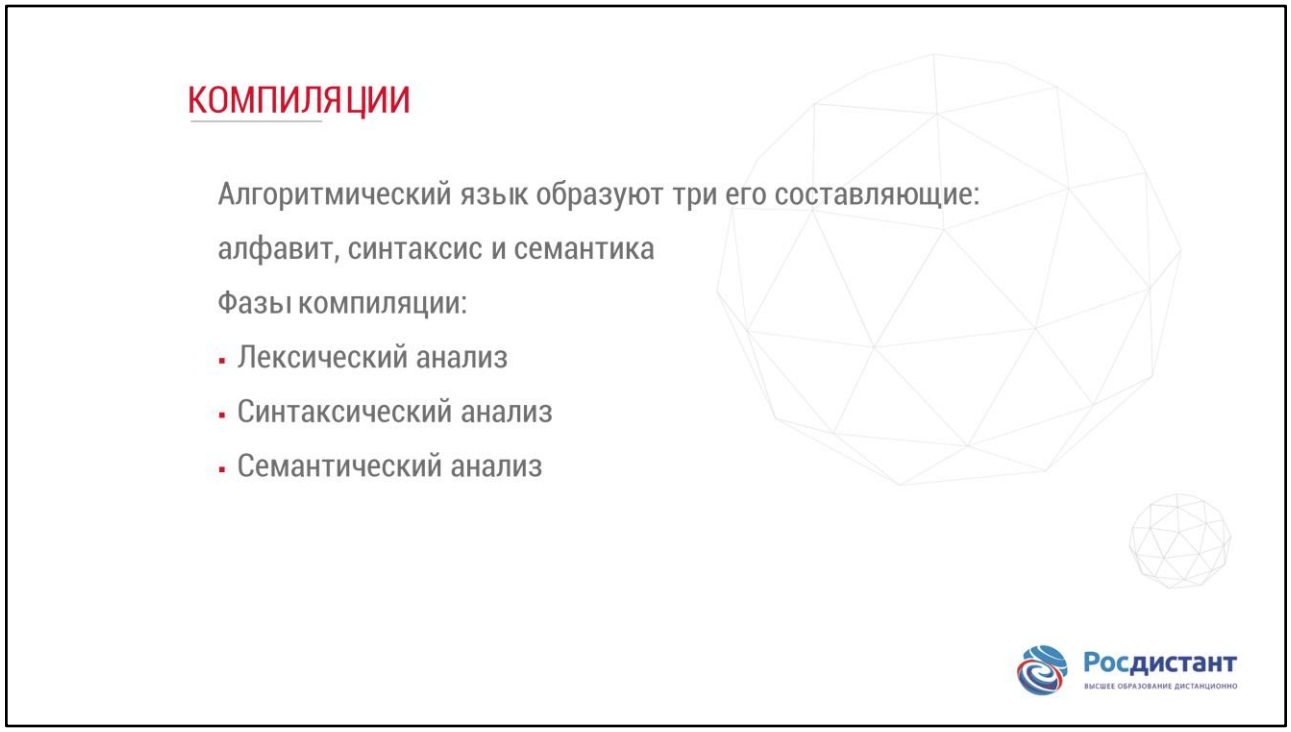
Интерпретаторы более гибкие, но менее эффективны, так как процесс интерпретации выполняется повторно при каждом запуске программы. Интерпретаторы бывают очень полезны, когда надо быстро проверить какую-то идею, для отладки или во время обучения новому языку.

Процесс интерпретации показан на слайде. Интерпретатор построчно читает

исходный код программы и выполняет инструкции, содержащиеся в текущей строке, потом переходит к следующей строке. Когда используется интерпретатор, он должен присутствовать все время для выполнения программы.

Любой язык программирования может быть компилируемым или интерпретируемым, однако, такие языки, как **C , C++** и **Pascal** - компилируются, в то время как скриптовые языки, такие, как **Perl** и **JavaScript** - интерпретируются.

Некоторые языки программирования, например, **Java** , могут как компилироваться, так и интерпретироваться.



Слайд 9

Обычный разговорный язык состоит из четырех основных элементов: символов, слов, словосочетаний и предложений. Алгоритмический язык содержит подобные элементы, только слова называют элементарными конструкциями, словосочетания - выражениями, предложения - операторами.

Входом компилятора служит программа на языке программирования. С точки зрения компилятора это просто последовательность символов. Задача первой фазы компиляции, лексического анализатора, заключается в разборе входной цепочки и выделении некоторых более крупных единиц, лексем. Примерами лексем являются основные ключевые слова, идентификаторы, константные значения и так далее. На этапе лексического анализа обычно также выполняются такие действия, как удаление комментариев.

Синтаксис - совокупность правил некоторого языка, определяющих формирование его элементов. Синтаксис задается с помощью правил, которые описывают понятия некоторого языка. Примерами понятий являются: переменная, выражение, оператор, процедура.

Именно иерархия объектов, а не то, как они взаимодействуют между собой, определяются через синтаксис. Например, оператор может встречаться только в процедуре, а выражение в операторе, переменная может состоять из имени и необязательных индексов и так далее. Синтаксис не связан с такими явлениями

в программе, как несоответствие типов или переменная с данным именем не определена. Этим занимается семантика.

Семантика - правила и условия, определяющие соотношения между элементами языка и их смысловыми значениями, а также интерпретацию содержательного значения синтаксических конструкций языка.

Текст исходной программы на языке высокого уровня представляет собой обычный тестовый файл. Для его чтения и превращения в последовательность машинных команд, прежде всего, выполняется синтаксический анализ текста программы.

Синтаксический анализатор - компонента компилятора, осуществляющая проверку исходных операторов на соответствие синтаксическим правилам и семантике данного языка программирования.

Слайд 10

Процесс решения задач на компьютере - это совместная деятельность человека и компьютера.

На долю человека приходятся этапы, связанные с творческой деятельностью – постановкой задачи, разработкой алгоритма вычислительного процесса, программированием задач и анализом результатов. На этом этапе участвует человек, хорошо представляющий предметную область задачи.

На долю персонального компьютера приходятся этапы обработки информации в соответствии с разработанным алгоритмом.

Языки программирования относятся к группе формальных языков, для которых в отличие от естественных языков однозначно определены синтаксис и семантика. Описание синтаксиса языка включает определение алфавита и правил построения различных конструкций языка из символов алфавита и более простых конструкций. Для этого обычно используют форму Бэкуса-Наура, сокращенно **БНФ** , синтаксические диаграммы.

Синтаксические диаграммы отображают правила построения конструкций в более наглядной форме.

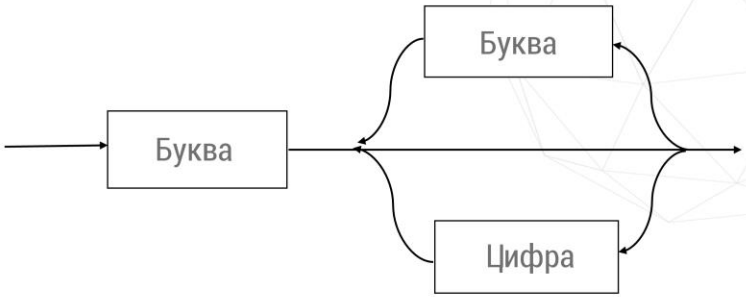
Представлена синтаксическая диаграмма, иллюстрирующая первые два правила описания конструкции – типа целое. Из диаграммы видно, что целое число может быть записано со знаком или без и включать произвольное количество

цифр.

Расширенная форма Бэкуса — Наура, **РБНФ** — формальная система определения синтаксиса, в которой одни синтаксические категории последовательно определяются через другие. Используется для описания контекстно-свободных формальных грамматик.
Предложена Никлаусом Виртом. Является расширенной переработкой форм Бэкуса - Наура, отличается от **БНФ** более ёмкими конструкциями, позволяющими при той же выразительной способности упростить и сократить в объёме описание.

ФОРМУЛА БЕКУСА-НУАРА (БНФ)

Синтаксическая конструкция <Идентификатор>



Росдистант
ВЫСШЕЕ ОБРАЗОВАНИЕ ДИСТАНЦИОННО

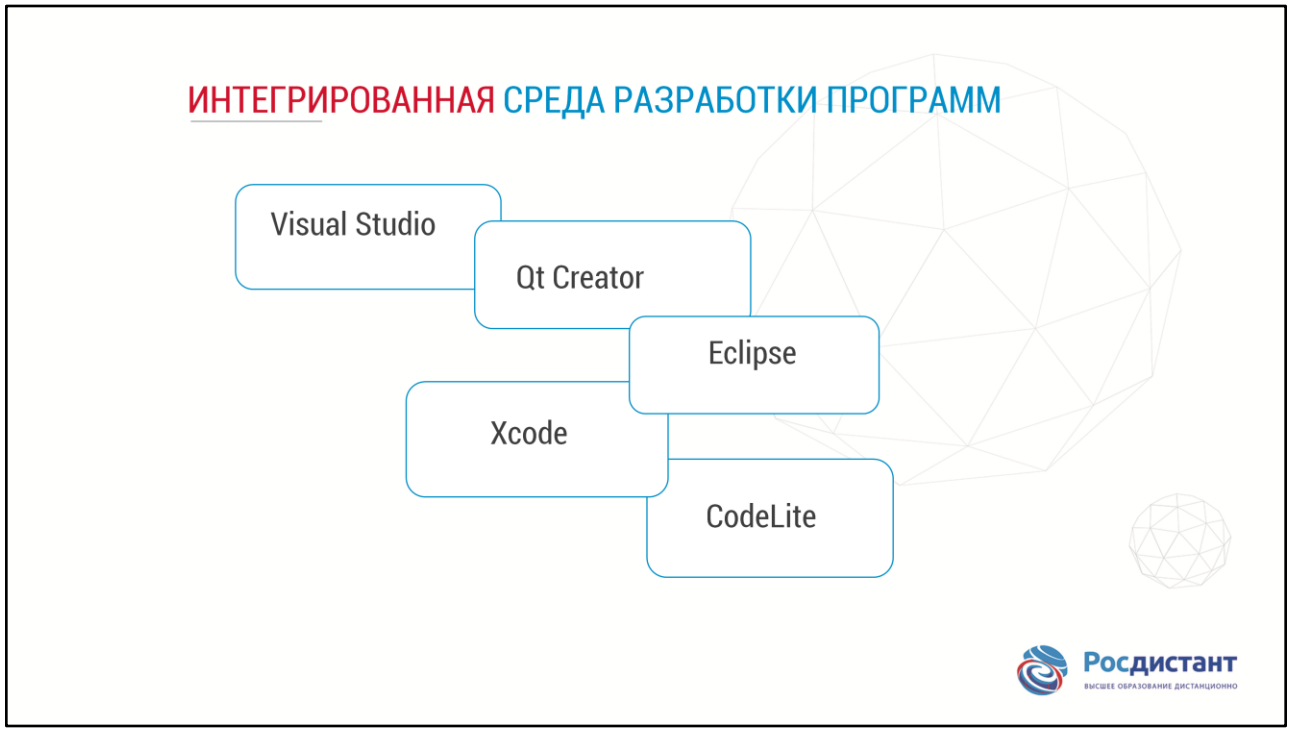
Слайд 11

Из символов алфавита в соответствии с правилами синтаксиса строят различные конструкции. Простейшей из них является конструкция – Идентификатор.

Конструкция Идентификатор используется во многих более сложных конструкциях для обозначения имен программных объектов, полей данных, процедур, функций и так далее.

Синтаксическая диаграмма идентификатора приведена на слайде. Семантику языка программирования закладывают в компилятор. Таким образом, синтаксически корректная программа, написанная на языке программирования, после преобразования ее в последовательность машинных команд обеспечит выполнение компьютером требуемых операций.

Такие же диаграммы существуют и для других объектов языка.



Слайд 12

Интегрированная среда разработки - комплексное средство, включающее всё необходимое программисту для создания программного обеспечения. Интегрированная среда разработки - это комплекс программ для разработки, компиляции, линкинга и отладки программ. На сегодняшний день существует множество интегрированных сред для **C++ ,** со своими особенностями, достоинствами и недостатками.

Линкер - это программа, которая производит компоновку: принимает на вход один или несколько объектных модулей и собирает по ним исполнимый модуль.

Перед выбором интегрированной среды необходимо определиться, для каких целей она нужна.

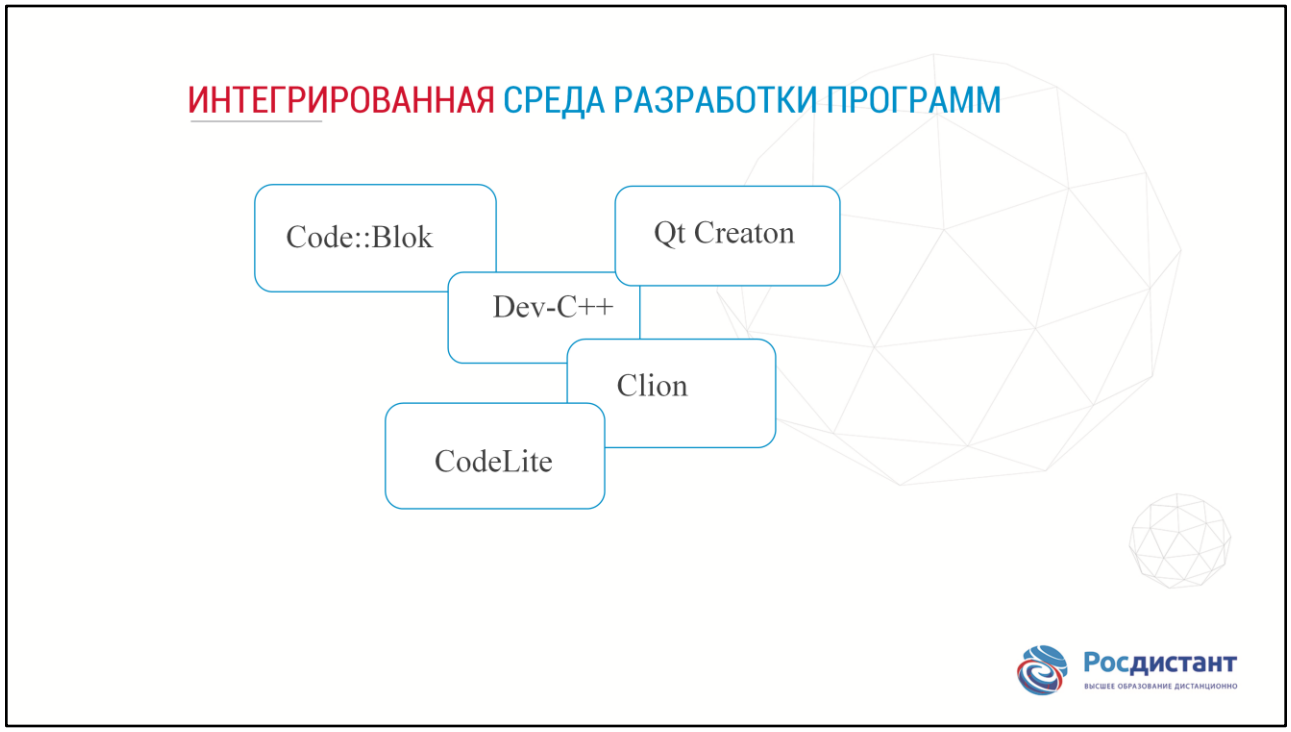
На слайде перечислены некоторые интегрированные среды для языка **C++ .**

«**Visual Studio** в основном известна для написания приложений. Это полный набор инструментов, позволяющий произвести точную отладку и настройку приложения. Есть как **Community** -версия, так и **PRO .**

Visual Studio предназначена не только для разработчиков на **C++** , но также поддерживает многие другие популярные языки, такие как **C# , Visual Basic**. **Visual Studio** предлагает множество функций:

- интеллектуальное автодополнение кода;
- дизайнер графических форм;
- простая в использовании навигационная система.

Вы можете использовать **Visual Studio** для разработки компьютерных программ для **Microsoft Windows** , а также веб-сайтов, веб-приложений и веб-сервисов.»



Слайд 13

«Codeblocks , – это еще одна свободная и открытая среда для **C++ .** Некоторые из функций **Codeblocks :**

- простая и быстрая установка;
- наличие портативной версии;
- встроенная возможность создания блок-схем.

Dev-C++ – это бесплатная интегрированная среда разработки с открытым исходным кодо, легкая среда, которой требуется всего на пару минут для установки. Это – лучшая среда разработки для новичков, в ней можно установить плагин для создания интрефейсов методом перетаскивания элементов. Некоторые из возможностей **Dev-C++** :

- малый вес;
- простая в использовании панель инструментов;
- автозавершение кода;
- простая установка.

CLion – платная программа, не имеющая бесплатной версии,

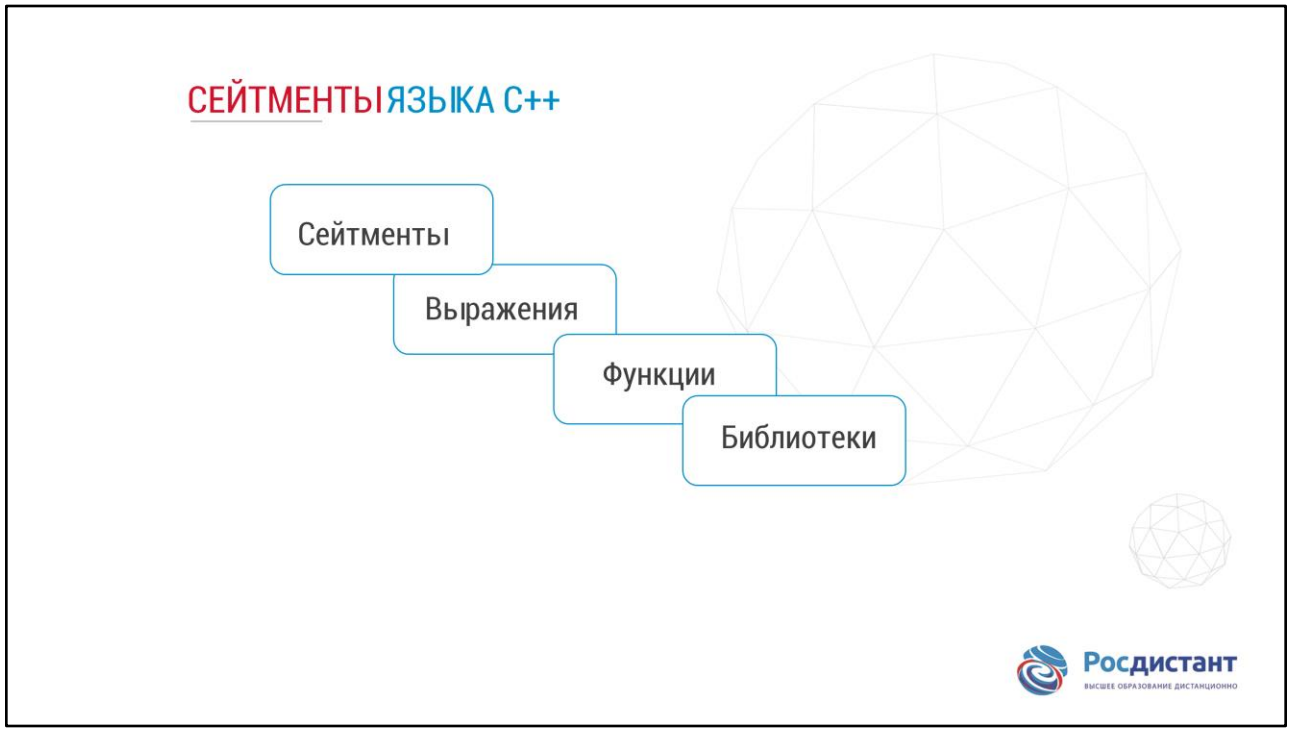
Возможности **CLion :**

- удобное создание визуализированных интерфейсов;

- наличие инструментов для удобного создания кода и отладки;
- возможность установки плагинов;
- поиск ошибок в коде в Live-режиме.»

В первую очередь выбор интегрированные среды зависит от уровня программирования. Для новичков лучшие среды разработки – это **Dev-C++** и **Codeblocks .**

«Конечно, современные интегрированные среды разработки предлагают программистам гораздо больше возможностей, чем входят в описанный выше необходимый минимум. Например, многие современные интегрированные среды являются визуальными - они позволяют создавать интерфейс программы с помощью мышки, точно в таком виде, в каком он предстанет потом пользователю. интегрированные среды, не являющиеся визуальными, требуют от программиста писать специальный код, ответственный за создание пользовательского интерфейса программы.»



Слайд 14

Познакомимся с таким понятием, как стейтмент. Компьютерная программа - это последовательность инструкций, которые сообщают компьютеру, что ему нужно сделать.

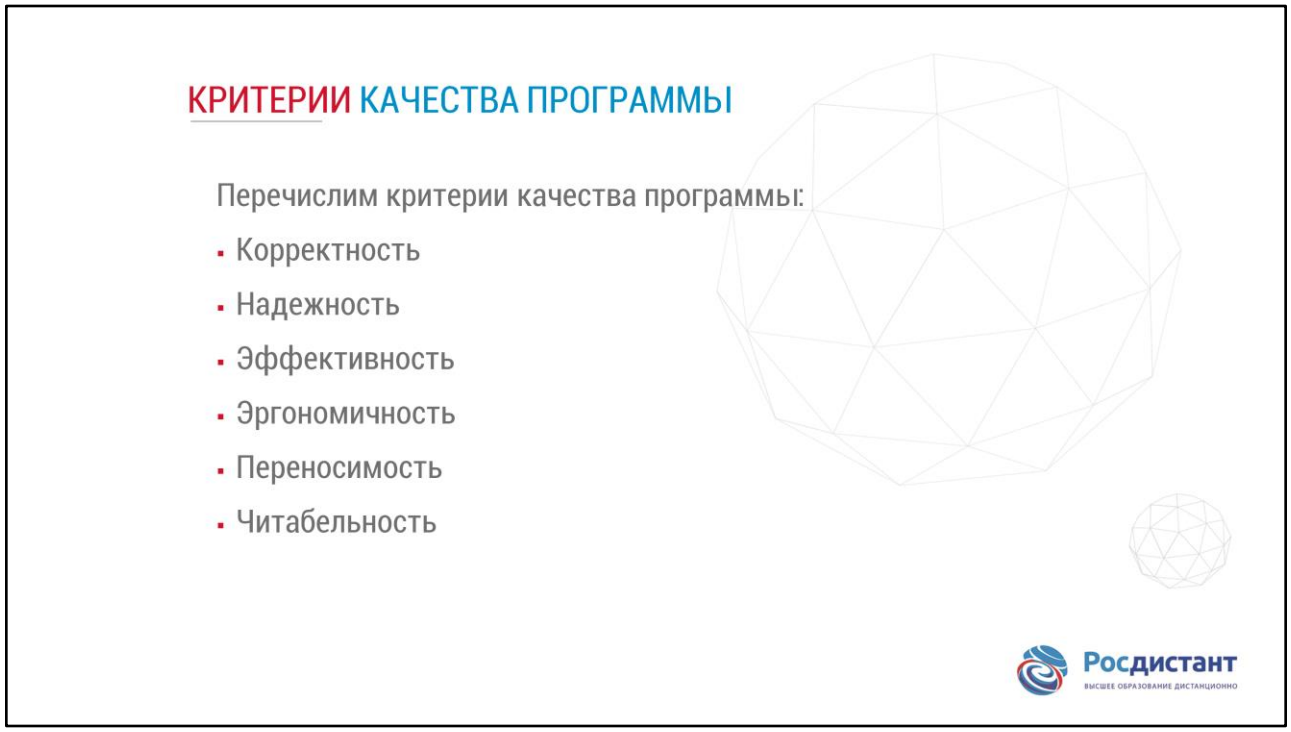
«Стейтмент - это наиболее распространенный тип инструкций в программах. Это и есть та самая инструкция, наименьшая независимая единица в языке **C ++** . Стейтмент в программировании - это то же самое, что и предложение в русском языке. Мы пишем предложения, чтобы выразить какую-то идею. В языке **C ++** мы пишем стейтменты, чтобы выполнить какое-то задание. Все стейтменты в языке **C ++** заканчиваются точкой с запятой.

Есть много разных видов стейтментов в языке C++. Рассмотрим самые распространенные из них:

- стейтмент объявления, **statement declaration .** Он сообщает компилятору, имя и тип переменной. В программировании каждая переменная занимает определенное число адресуемых ячеек в памяти в зависимости от её типа. Минимальная адресуемая ячейка — байт. Все переменные в программе должны быть объявлены, прежде чем использованы;
- стейтмент присваивания, **assignment statement** ;
- стейтмент вывода, **output statement .** Мы выводим значение переменной на экран.

Компилятор может также обрабатывать выражения. Это математический объект, который создается для проведения вычислений и нахождения результатов. Выражения могут содержать цифры, буквенные переменные, операции, математические функции.

В языке **C ++** стейтменты объединяются в блоки - функции. Функция - это последовательность стейтментов. Каждая программа, написанная на языке **C ++ ,** должна содержать главную функцию **main .** Именно с первого стейтмента, находящегося в функции **main** и начинается выполнение всей программы. Функции, как правило, выполняют конкретное задание. Например, может содержать стейтменты, которые определяют большее из заданных чисел, или вычислять среднюю оценку студентов по какой-либо дисциплине.»



Слайд 15

На слайде представлены критерии качества программы.

- «Корректность – очевидно, что программа должна работать правильно, иначе нет смысла ее писать.
- Надежность – программа не должна зависать или зацикливаться при любых исходных данных.
- Эффективность – программа должна использовать, по возможности, минимальное количество ресурсов по памяти, хотя в настоящее время это стало менее актуально. Минимализация времени работы программы - это проблема остается актуальной, особенно при обработке больших массивов данных.
- Эргономичность – удобство для пользователя. Не забывайте, что первым пользователем Вашей программы будете Вы сами.
- Переносимость – программа должна работать не только на Вашем компьютере, но и на других!
- Читабельность – удобство для программиста. К сожалению, по прошествии времени, программа забывается, и давно написанную Вами программу невозможно прочитать как книгу, Вы должны заново принимать все когда-то принятые решения. Поэтому нет смысла создавать себе дополнительные

трудности в виде плохо структурированного и плохо читаемого текста. Для решения этой проблемы существует ряд принципов написания текста программы.

1. Не стоит писать, как слишком длинные строки – они уходят за пределы экрана по ширине, так и слишком короткие строки – они увеличивают длину не только всей программы, но и каждого отдельного блока, который Вы в этом случае не можете охватить взглядом и оценить, что он делает.
2. Принято вложенные блоки писать со смещением вправо.
3. Не прячьте фигурные скобки в конец строки – в этом случае их трудно найти и определить начало и конец блока.»

Слайд 16

«Среди современных языков программирования язык **C++** является одним из наиболее распространенных, это язык программирования общего назначения, цель которого – сделать работу серьёзных программистов более приятным занятием. Язык C++ обеспечивает гибкие и эффективные средства определения новых типов. Язык **C++** хорошо зарекомендовал себя эффективностью, лаконичностью записи алгоритмов, логической стройностью программ. Во многих случаях программы, написанные на языке **C++** , сравнимы по скорости с программами, написанными на Ассемблере, при этом они более наглядны и просты в сопровождении.

Перечислим некоторые особенности языка **C++** ,.

В языке реализован ряд операций низкого уровня. Некоторые из таких операций напрямую соответствуют машинным командам, например, поразрядные операции или операции **++ и - -** .

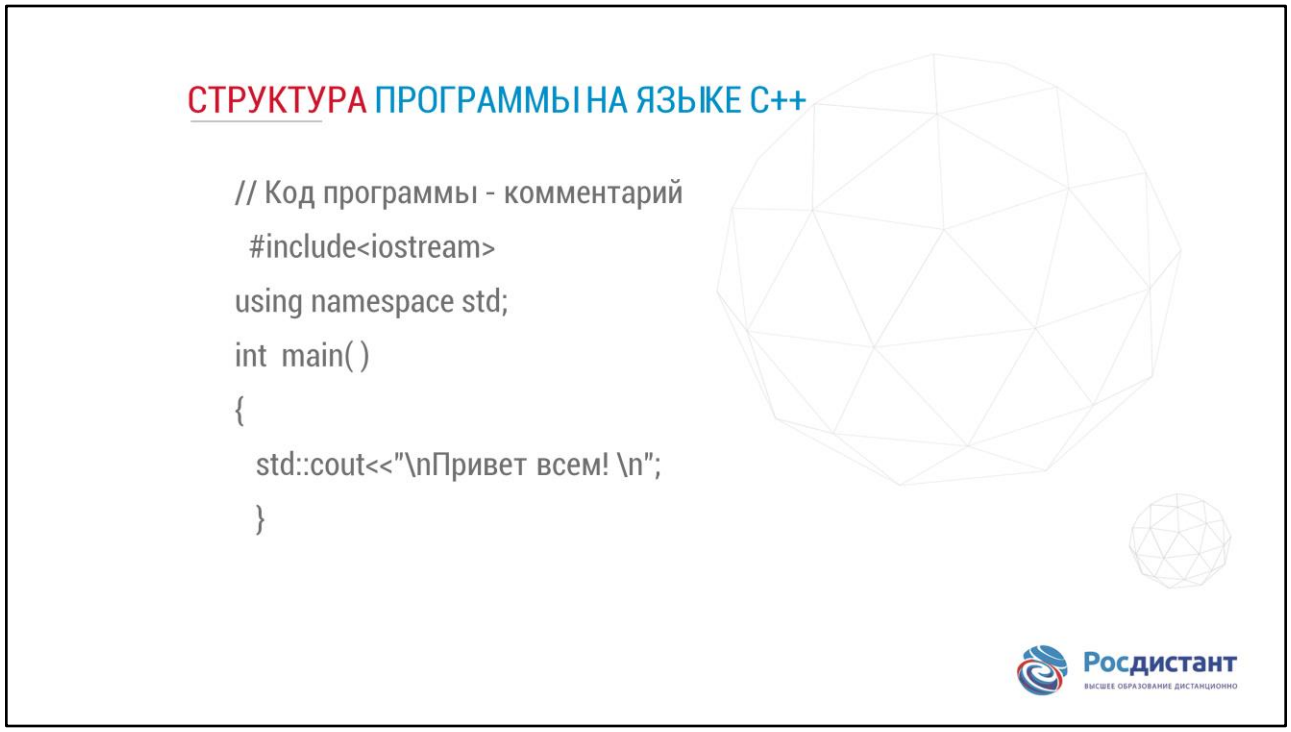
Базовые типы данных языка отражают те же объекты, с которыми приходится иметь дело в программе на Ассемблере – байты, машинные слова и так далее.»

Слово программа происходит от двух греческих слов: пред и запись, в переводе означающий предписание, то есть означает - заданную последовательность действий.

Программа на алгоритмическом языке записывается с помощью символов,

образующих алфавит языка. Он включает заглавные и прописные буквы латинского алфавита, арабские цифры и некоторые специальные символы.

Остальные символы могут быть использованы только в символьных строках, символьных константах и комментариях. На языке **C++** заглавные и прописные буквы различаются. Особенно это надо учитывать при составлении имен переменных, идентификаторов.



Слайд 17

Программа на языке **C++** состоит из директив препроцессора, описаний и определений, набора функций, среди которых обязательно есть одна с именем **main** , и именно с нее начинается выполнение программы.

На слайде представлена простейшая программа на языке **C++** . Программа начинается с комментарий – первая строка программы. В комментариях можно использовать буквы русского алфавита.

Вторая строка - содержит команду препроцессора. Препроцессор - это специальная программа, являющаяся частью компилятора языка **C++ .** Она сообщает транслятору, что надо включить в программу описания, необходимые для работы стандартных потоков ввода-вывода, которые находятся в **iostream** .

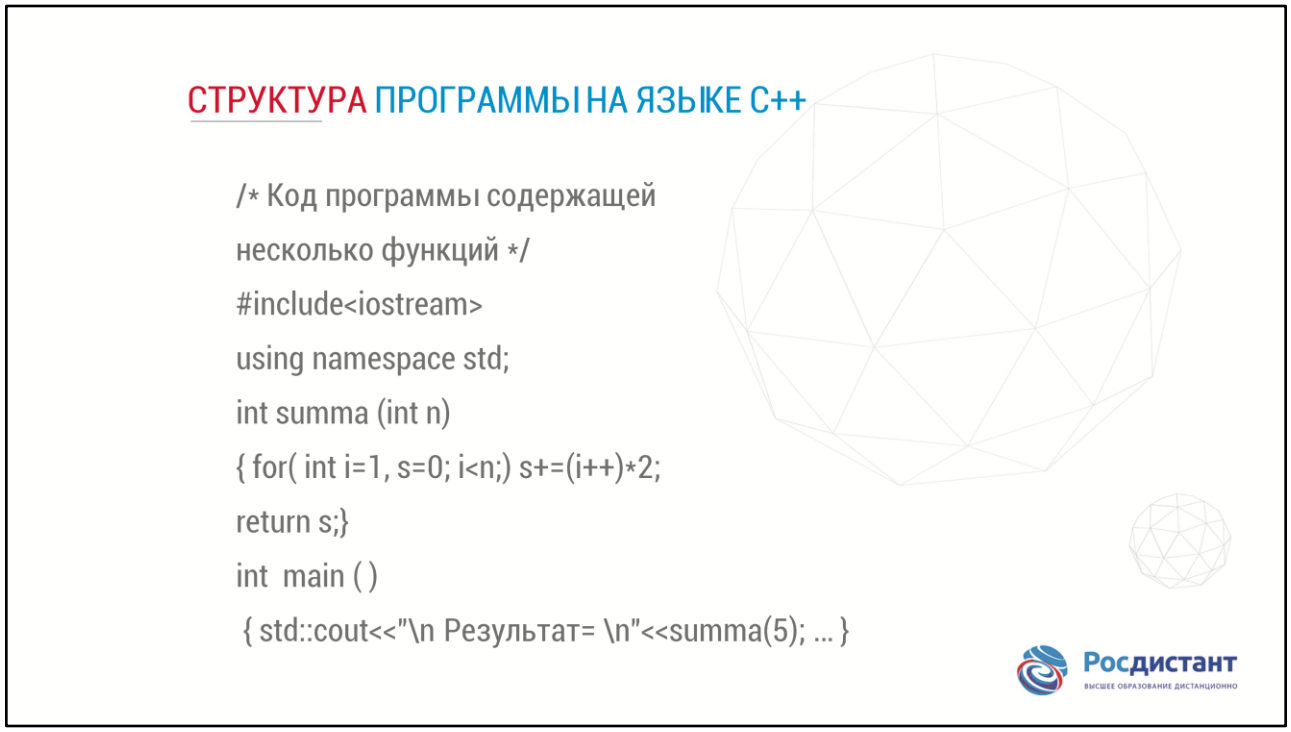
Команда препроцессора **include** начинаются с символа решетка. Используя угловые скобки, мы сообщаем компилятору, что подключаемый заголовочный файл предоставляется Стандартной библиотекой.

Третья строка. Фраза, указанная в этой строке, означает - используя пространство имен **std** .

«Запись **using namespace std** говорит о том, что будет использоваться пространство имен **std ,** в котором определена вся стандартная библиотека языка. Здесь пространство имен – это область действия объектов программы, в том числе переменных. В области имен **std ,** описаны стандартные потоки ввода

и вывода. Если не объявлять используемое пространство имен, нужно было бы явно указывать его при любом обращении к этим потокам.»

Четвертая строка - заголовок функции с именем **main** . Каждая программа на **C++** должна содержать только одну функцию с таким именем. Пустые круглые скобки – нужны для соблюдения синтаксиса обращения к функциям. Обычно в них помещают список формальных параметров. Пятая строка выдает приветствие – Привет всем.



Слайд 18

На слайде представлен фрагмент программы, содержащей функцию с именем **summa .**

Функция – это поименованная часть программы, которая может вызываться из других частей программы столько раз, сколько необходимо. Первая часть определения функции задает ее имя, тип возвращаемого значения, если оно есть, а также типы и имена формальных параметров. Значение возвращается из функции с помощью оператора **return .**

Тело любой функции, включая главную функцию **main ,** это последовательность описаний, определений, операторов, заключенных в фигурные скобки. Каждое описание, определение или оператор заканчивается символом точка с запятой. Переменные, описанные в теле любой функции, являются локальными и доступны в теле только данной функции. Переменные, которые описаны вне всяких функций, являются глобальными и доступны в каждой функции.

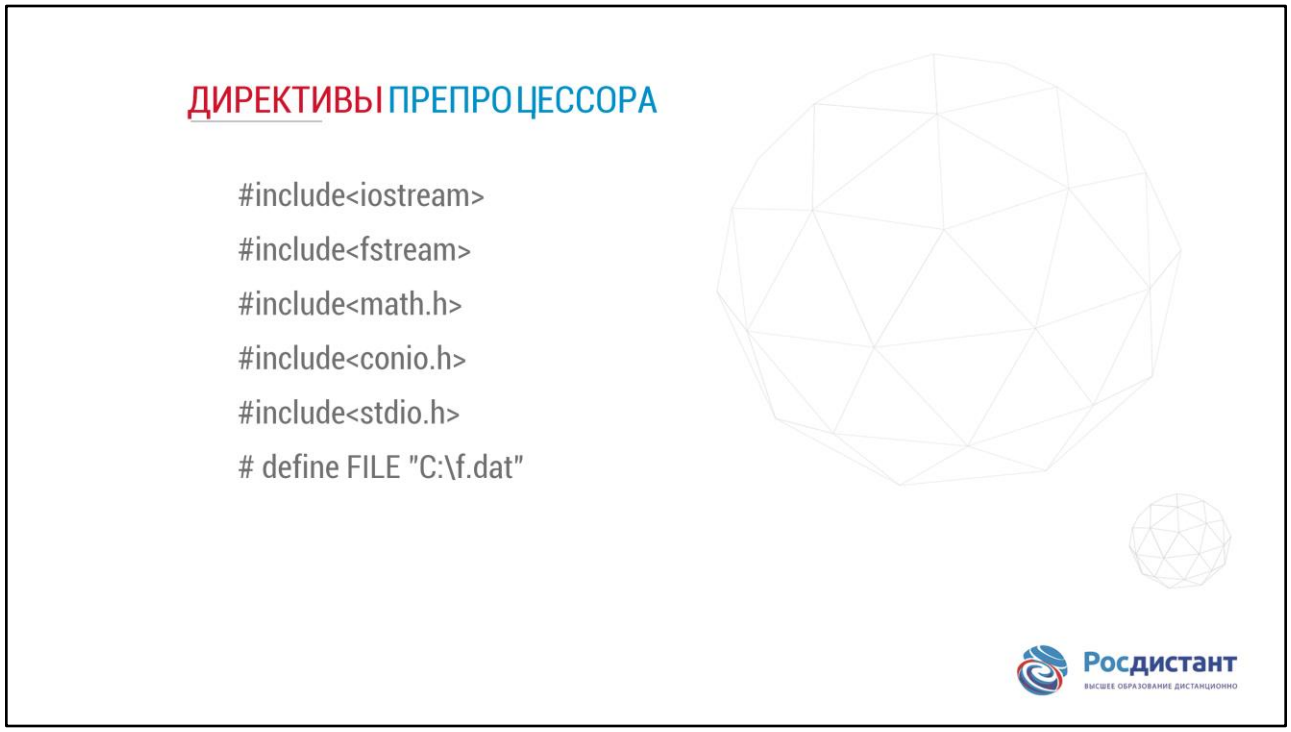
«Комментарии нужно писать так, чтобы человек, который не имеет ни малейшего представления о том, что делает ваш код - смог в нем разобраться. Однострочные комментарии - это комментарии, которые пишутся после символов // и пишутся в отдельных строках и всё, что находится после этих символов комментирования, - игнорируется компилятором. Как правило, однострочные комментарии используются для объяснения одной строчки

кода.»

Многострочные комментарии начинаются с символов **/*** и заканчиваются символами ***/** . Еще раз напомним, что комментарии компилятором не обрабатываются.

Функция вывода информации на экран – **cout** .

Два рядом стоящих знака меньше – означает поместит в выходной поток. В выходной поток помещается всё, что находится справа от этих символов. В данной программе – это строковая константа, заключённая в кавычки. Строковая константа может содержать любые символы, включая и те, которые не изображаются на экране, а являются управляющими символами, например - **\n** – означает переход к началу следующей строки экрана.



Слайд 19

Препроцессор – это специальная программа, являющаяся частью компилятора языка **C++** . Она предназначена для предварительной обработки текста программы. Препроцессор позволяет включать в текст программы файлы. Работа препроцессора осуществляется с помощью специальных директив, указаний. Они отмечаются знаком решетка.

Рассмотрим некоторые директивы препроцессора:

- **include** — вставляет текст из указанного файла, подключаются стандартные библиотеки, для использования стандартных функций.

На слайде приведены примеры подключения различных заголовочных файлов:

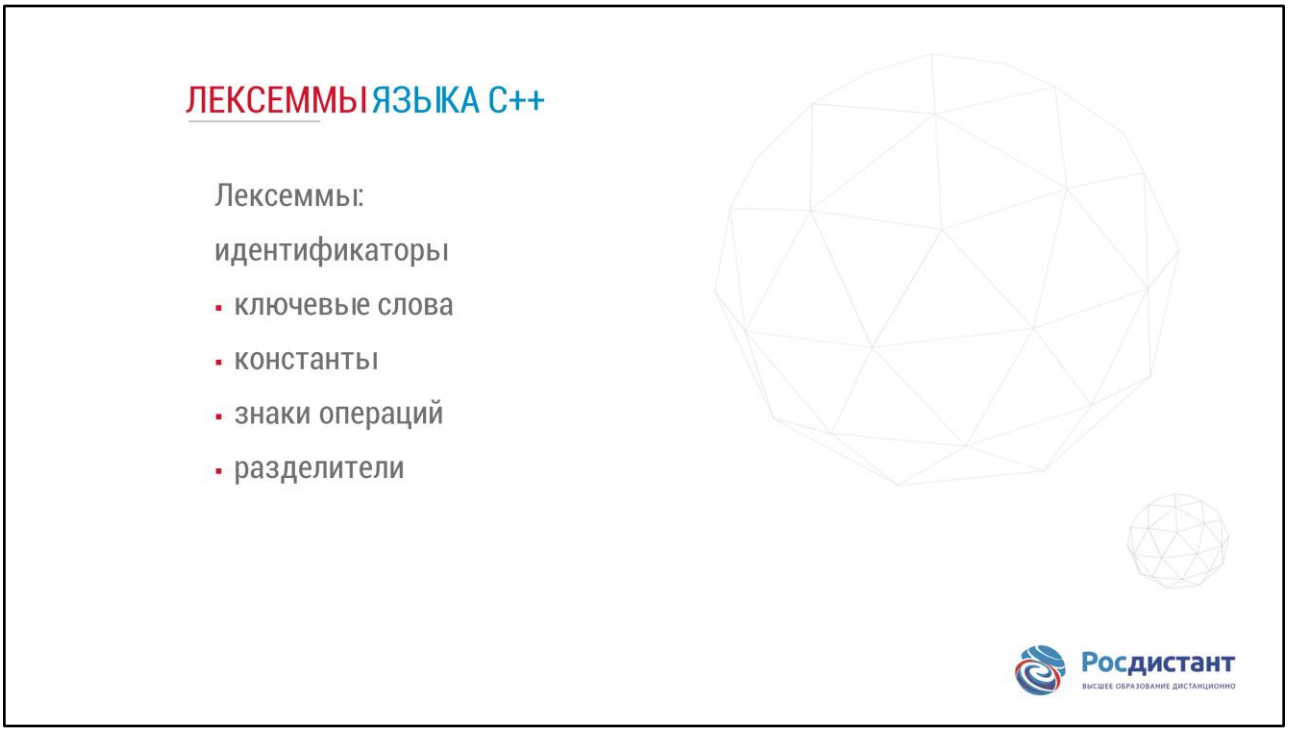
- **iostream** - для работы с потоками ввода-вывода;
- **fstream** - для работы с файлами данных;
- **math** -для использования математических функций;
- **define** - подстановка в тексте.

Директива **include** позволяет включать в текст программы указанный файл. Если файл является стандартной библиотекой и находится в папке компилятора, он заключается в угловые скобки, если нет, он указывается в кавычка. Для файла, находящегося в другом каталоге необходимо в кавычках указать полный путь.

Директива **define** определяет подстановку в тексте программы. Она

используется для определения символических констант. Указывается имя подстановки и текст подстановки. Все вхождения имени заменяются в коде на текст подстановки. В примере на слайде директива **define** использована для ввода символической константы **FILE ,** связанной с именем файла **f.dat.** Указан и путь доступа к файлу. В дальнейшем в программе, при обращении к файлу вместо имени файла используется символическая константа.

Имена символических констант рекомендуется записывать прописными буквами, чтобы зрительно отличать их от имен переменных и функций.



Слайд 20

Из символов языка формируются лексемы – это идентификаторы, ключевые слова, константы, знаки операций. Границы лексем определяются другими лексемами, такими, как разделители или знаки операций.

Идентификаторы используются для описания имен переменных, имен функций, пользовательских типов данных - структур, констант и т.д.

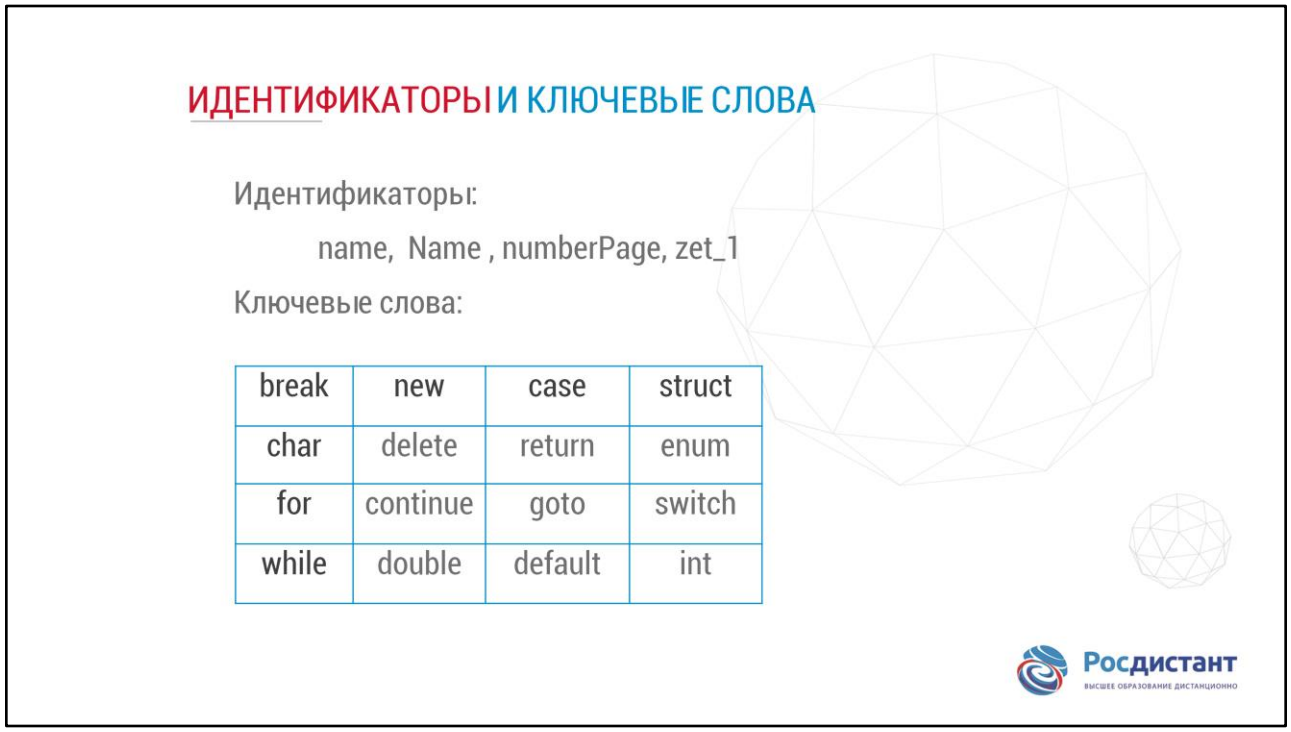
Идентификатор — это имя программного объекта. В идентификаторе могут использоваться латинские буквы, цифры и знак подчеркивания.

Ключевые зарезервированные слова в языке С ++ предназначены для специального использования. Ключевые слова могут быть использованы как имена стандартных библиотечных функций, как имена операторов, и не могут быть использованы в виде идентификаторов. На слайде приведены некоторые ключевые слова.

Не следующем слайде мы подробно рассмотрим привила составления идентификаторов в программе.

Константами называют неизменяемые величины. Различаются целые, вещественные, символьные и строковые константы. Компилятор, выделив константу в качестве лексемы, относит ее к одному из типов по ее внешнему виду.

Знак операции — это один или более символов, определяющих действие над операндами. Выражение - это последовательность операндов, связанных знаками арифметическими или логическими операций. Операндом может быть переменная, числовая константа, стандартная функция. Внутри знака операции пробелы не допускаются. Операции делятся на унарные, бинарные и тернарную по количеству участвующих в них операндов. Большинство стандартных операций может быть переопределено, перегружено. Перегрузка операций будет рассмотрена позже.



Слайд 21

«В языке С++ все идентификаторы, имена переменных, функций, классов и так далее, должны быть уникальными. Если в вашей программе находятся два одинаковых идентификатора, то будьте уверены, что ваша программа не скомпилируется: вы получите ошибку конфликта имен.

Как только программы становятся больше, то и идентификаторов используется больше. Следовательно, вероятность возникновения конфликта имен значительно возрастает. Язык **C ++** предоставляет достаточно механизмов для предотвращения возникновения конфликтов имен, например, локальная область видимости или пространства имен. Самый простой способ сообщить компилятору, что определенный идентификатор находится в определенном пространстве имен — использовать оператор разрешения области видимости.»

Идентификаторы используются для описания имен переменных, функций, пользовательских типов данных - структур, констант и так далее.

На слайде приведены примеры имен переменных. Прописные и строчные буквы на языке **C ++** различаются.

Мы можем определять идентификаторы любыми словами, то есть любыми именами. Есть несколько общих правил, которые необходимо соблюдать:

- идентификатор не может быть ключевым словом. Ключевые слова зарезервированы;

- идентификатор может состоять только из букв нижнего или верхнего регистра, цифр или символов подчёркивания, все другие символы и пробелы запрещены;

- идентификатор должен начинаться с буквы, идентификатор не может начинаться с цифры или знака подчеркивания.

«Как правило, на **C ++** имена переменных и функций пишутся буквами в нижнем регистре. Имена идентификаторов, которые начинаются с заглавной буквы, используются как имена структур, классов или перечислений, о чем мы поговорим позже.

Если имя переменной или функции состоит из нескольких слов, то есть два варианта: разделить подчёркиванием или использовать - принцип, когда несколько слов пишутся слитно, без пробелов, и каждое новое слово пишется с заглавной буквы. **CamelCase ,** что в переводе означает верблюжий стиль, получил своё название из-за заглавных букв, которые напоминают верблюжьи горбы.»

На слайде приведены некоторые ключевые слова. Язык С++ имеет зарезервированный набор из 84 слов для собственного использования, даже в версии **C++ 17**. Эти слова называются ключевыми словами, каждое из которых имеет свое особое значение. В ходе изучения нашего курса мы будем знакомиться с ключевыми словами и их назначением.

=

Слайд 22

Литералы — это ограниченная последовательность символов алфавита языка, представляющая собой изображение фиксированного, неизменяемого объекта.

В программе можно использовать литералы, которые могут быть следующих типов:

- целые;
- вещественные, с фиксированной или с плавающей точкой;
- символьные;
- строковые.

Целые литералы могут быть десятичные, восьмеричные и шестнадцатеричные. На слайде приведены примеры литералов, констант различных типов.

Десятичные литералы – это, последовательность цифр со знаком плюс или минус.

Восьмеричные литералы начинаются с литеры ноль, а шестнадцатеричные – с двух литерал, ноль и икс, причем икс может быть прописная или заглавная.

В зависимости от значения литерала компилятор присваивает ей тот или иной тип, а именно – **char , int** или **long int** .

Литералы можно использовать в арифметических выражениях. Помимо

неименованных констант, в выражениях можно использовать и именованные литералы.

ЛИТЕРАЛЫ

Вещественные	Примеры
▪ фиксированной точкой	15.63 -0.55 .55
▪ с плавающей точкой	0.85 e2 35. e-5 -0.55 e-3

Росдистант
ВЫСШЕЕ ОБРАЗОВАНИЕ ДИСТАНЦИОННО

Слайд 23

Вещественные литералы бывают с фиксированной или с плавающей точкой.

Числа с плавающей точкой - это вещественные числа, в которых число хранится в виде мантиссы и порядка, то есть показателя степени. При этом число с плавающей точкой имеет фиксированную относительную точность и изменяющуюся абсолютную.

Реализация математических операций с числами с плавающей запятой в вычислительных системах может быть как аппаратная, так и программная.

Число с плавающей точкой состоит из следующих частей:

- знак мантиссы, который указывает на отрицательность или положительность числа;
- мантисса – выражает значение числа без учёта порядка;
- знак порядка;
- порядок, то есть выражает степень основания числа, на которое умножается мантисса.

По умолчанию компилятор присваивает вещественному числу тип **double .** В вещественных литералах используется десятичная точка, разделяющая целую и дробную части.

ЛИТЕРАЛЫ

Символьные	Примеры
одиночные символьные	'z ', 'A', 'П', 'Я',
управляющие символы	'\ a' - звуковой сигнал '\ n' - перевод строки '\ t' - горизонтальная табуляция '\ v' - вертикальная табуляция
Строковые	"Это строковая константа"

Росдистант
ВЫСШЕЕ ОБРАЗОВАНИЕ ДИСТАНЦИОННО

Слайд 24

Символьные литералы — это один символ, например - **z** , как показано на слайде.

В качестве символьных литерал также могут использоваться управляющие коды, не имеющие графического представления. При этом код управляющего символа начинается с символа - обратный слеш. На слайде приведены некоторые управляющие символы.

«Все символьные литералы имеют тип **char** и занимают в памяти один байт. Значением символьной константы является числовое значение её внутреннего кода. Коды символов представлены в международной таблице **ASCII** кодов.

На языке **C++** допустимы и строковые литералы. Строковые литералы представляет собой последовательность любых символов, заключенная в двойные кавычки. Кавычки не входят в строку, а лишь ограничивают её.

Технически строковые литералы представляет собой массив символов, и по этому признаку могут быть отнесены к разряду сложных объектов языка.»

Строковые литералы мы подробно рассмотрим в следующих темах нашего курса.

ТИПЫ ДАННЫХ

Типы	Байт	Диапазон значений
целочисленный (логический) тип		
bool	1	0 / 255
целочисленный (символьный) тип		
char	1	0 / 255
вещественный тип		
float	4	2 147 483 648.0 / 2 147 483 647.0
double	8	- 9 223 372 036 854 775 808.0 / 9 223 372 036 854 775 808.0

Росдистант
ВЫСШЕЕ ОБРАЗОВАНИЕ ДИСТАНЦИОННО

Слайд 25

«Объем памяти, который использует переменная, зависит от типа данных этой переменной. Так как мы, как правило, получаем доступ к памяти через имена переменных, а не через адреса памяти, то компилятор может скрывать от нас все детали работы с переменными разных размеров.

Есть несколько причин по которым полезно знать, сколько памяти занимает определенная переменная, определенного типа.

Компьютеры имеют ограниченное количество свободной памяти. Каждый раз, когда мы объявляем переменную, небольшая часть этой свободной памяти выделяется до тех пор, пока переменная существует. Фактический размер переменных может отличаться на разных компьютерах, поэтому для его определения используют операцию **sizeof .**»

Все переменные, используемые в программе на языке **C++** , должны быть обязательно описаны. При описании переменных указывается имя переменной и тип, определяющий объем памяти, выделяемый под хранение переменной. Описывать переменные можно в любом месте программы, но до первого обращения к этой переменной.

В таблице на слайде представлены основные типы данных языка **C++ .**

Логический тип **bool** используется исключительно для хранения результатов логических выражений, которые могут принимать одно из двух значений **true ,**

что означает истина, или **false ,** что означает ложь. Этот тип переменных часто называют булевыми переменными.

Константе **true** эквивалентны все числа от одного до двухсот пятидесяти пяти включительно, тогда как константе **false** эквивалентно только одно целое число - ноль.

Тип данных **char** – это, символьный тип данных, который используется для представления символов. То есть, каждому символу соответствует определённое двоичное число из диапазона от нуля (0) до двухсот пятидесяти пяти. Для представления символов в C++ типу данных char отводится один байт, в одном байте – восемь бит, с помощью которых можно закодировать 256 символов. Коды символов представлены в таблице **ASCII** кодов.

Типы **float** и **double** используются для представления вещественных чисел.

Слайд 26

«Диапазон целочисленной переменной определяется двумя факторами: её размером, который измеряется в байтах, и её знаком, который может быть **signed** или **unsigned .**

Целочисленный тип **signed** , со знаком, означает, что переменная может содержать как положительные, так и отрицательные числа.

Целочисленный тип **unsigned** , без знака, может содержать только положительные числа.

Если попытаться использовать значение, которое находится вне диапазона значений определенного типа данных, то произойдет переполнение. Переполнение случается из-за того, что переменной не было выделено достаточно памяти для их хранения.»

На слайде приведены типы данных для представления целых чисел**.** Они характеризуются определенным размер занимаемой памяти в байтах и диапазоном принимаемых значений. Напоминаем, в зависимости от компилятора, размер занимаемой памяти и диапазон принимаемых значений могут изменяться.

Диапазон принимаемых значений меняется в случае, если тип данных объявляется с приставкой **unsigned** - без знака. Приставка **unsigned** говорит о том, что тип данных не может хранить знаковые значения и диапазон

положительных значений увеличивается в два раза.

Преобразования типов нужны в **C++** потому, что арифметические операции со смешанными типами являются нормой для языков, используемых в числовых задачах.

«Синтаксически тип **void** является фундаментальным типом. Его можно использовать только как часть сложного типа, так как объектов типа **void** не существует. Этот тип используют для указания на то, что функция не возвращает значения. Подробнее этот тип мы рассмотрим в следующих темах.»

Введение в языки программирования C и C++ | Уроки C++ - Ravesli / https://ravesli.com/urok-2-vvedenie-v-yazyki-programmirovaniya-c-i-s/

Т, А. Павловская С/С++ Программирование на языке высокого уровня/http://cph.phys.spbu.ru/documents/First/books/7.pdf

Введение в программирование | Уроки C++ - Ravesli/https://ravesli.com/urok-1-vvedenie-v-programmirovanie/

Технология программирования - Информатика, автоматизированные информационные технологии и системы/ https://studref.com/441961/informatika/tehnologiya_programmirovaniya

Бьерн Страуструп. Язык программирования С++ 11/ https://vk.com/doc-145125017_450056359

Язык СИ++ Учебное собие / http://5fan.ru/wievjob.php?id=4301

А.А. Шелупанов, В.Н. Кирнос ИНФОРМАТИКА. БАЗОВЫЙ КУРС Учебник в четырех частях. /https://edu.tusur.ru/publications/521/download