MINISTRY OF EDUCATION AND SCIENCE OF UKRAINE

Odessa I.I.Mechnikov national university

C++ Glossary

Термінологічний словник мови програмування C++

For the first and second year students of Information Technologies, Mechanics and Applied Mathematics Departments of IMEM

Odessa - 2015

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Одеський національний університет імені І.І. Мечникова

Термінологічний словник мови програмування С++

для студентів І та ІІ курсів Інституту математики, економіки і механіки, що навчаються за фахом «Інформаційні технології», «Механіка» та «Прикладна математика»

О.А. Румянцева

Одеса – 2015

УДК [811.11+811.161.1]’374.3:004.43
ББК 81.432.1-43
Р 865

Рекомендовано до друку НМК факультету романо-германської філології
Одеського національного університету імені І.І. Мечникова
Протокол № 8 від 16 березня 2015 року

Укладач: **Румянцева Олена Анатоліївна,** к.філол.н., ст. викладач кафедри іноземних мов природничих факультетів ОНУ імені І.І. Мечникова

Рецензенти: **Петлюченко Наталя Володимирівна,** д. філол. н., професор, завідувач кафедри германських і романських мов національного університету «Одеська юридична академія»;
Халілова-Костюк Наталя Олегівна, к. пед. н., доцент кафедри іноземних мов природничих факультетів ОНУ імені І.І. Мечникова;
Рачинська Алла Леонідівна, к. фіз.-мат. н., доцент кафедри теоретичної механіки, декан факультету прикладної математики та інформаційних технологій ОНУ імені І.І. Мечникова.

Румянцева О.А.

Р 865 Термінологічний словник мови програмування С++ (І та ІІ курсів Інституту математики, економіки і механіки, що навчаються за фахом «Прикладна математика» та «Інформаційні технології») = C++ Glossary (for the first and second year students of Information Technologies, Mechanics and Applied Mathematics Departments of IMEM) / О.А. Румянцева, ОНУ імені І.І. Мечникова. – Одеса, 2015. – 31 с.

Термінологічний словник мови програмування С++ створений для студентів І та ІІ курсів Інституту математики, економіки і механіки, що навчаються за фахом «Інформаційні технології», «Механіка» та «Прикладна математика», ОНУ імені І.І. Мечникова, для аспірантів, викладачів, які викладають С++, а також користувачів ПК, що вивчають С++. Словник містить найбільш використовуваних термінів у сфері інформаційних технологій і програмування. До нього увійшли не лише сталі терміни, але і найбільш популярні неологізми. Визначення термінів російською мовою відрізняються простотою і ясністю, тому вони є зрозумілі не лише фахівцям ІТ-сфери, але і широкому колу користувачів ПК. Словник призначений для розширення лексичного запасу фахівців в сфері інформаційних технологій, а також у навчальному процесі для вдосконалення знань студентів вузів.

УДК [811.11+811.161.1]’374.3:004.43
ББК 81.432.1-43

C++ Glossary
Терминологический словарь языка программирования C++
Meta-linguistic terms (металингвистические термины)

C++ - это язык программирования высокого уровня, разработанный Бьёрном Страуструпом (Bjarne Stroustrup) в AT&T Bell Laboratories (Муррей-Хилл, Нью-Джерси) в 1983 г. Объединяет возможности языка Си с ООП. Стандартизован ISO в 1998 г. (ISO/IEC 14882). Широко распространён, используется в системном и прикладном программировании. Нынешняя версия этого стандарта - ISO/IEC 14882:2003. Имеется множество доступных через Интернет компиляторов с C++.

1. **Object (объект)** - это то, с чем производится действие. Это может быть документ, открываемый в окне браузера или само окно браузера, или какая-то часть документа. Проще - это может быть любая виртуальная "вещь", которую используют на странице. Можно использовать стандартные объекты JavaScript, или придумывать и создавать их самим. Кроме того, программа может создавать объекты сама в процессе своей работы "для внутреннего использования".
2. **Property (свойство)** - каждый объект имеет свои свойства. Один и тот же объект может обладать многими свойствами: дом может быть большим и маленьким, синим и красным. Разные объекты могут обладать одинаковыми свойствами: дерево, так же, как и дом, может быть большим и маленьким, синим и красным... Большинство свойств объекта можно изменять, воздействуя на них через методы. (Собственно, это как раз то, что нужно: изменять свойства, то есть внешний вид объекта на странице)
3. **Method (метод объекта)** - это действие или способ, при помощи которого можно изменять определенные свойства объекта, то есть управлять этими объектами, а также в некоторых случаях менять их содержимое.
4. **Event (событие)** - это все, что случилось: открытие окна, загрузка в него документа, клик клавишей мышки или просто перемещение курсора по экрану, нажатие клавиши на клавиатуре - это все события, и они могут инициировать запуск больших и маленьких программ.
5. **Operator (оператор)** – это команда, инструкция для компьютера. Встретив в программе тот или иной оператор, машина четко его выполняет. Каждый язык программирования имеет свой, четко оговоренный набор операторов. В отличие от методов, свойств и объектов, мы не можем создавать свои операторы: интерпретатор языка не поймет их и не сможет четко перевести машине, чего мы от нее добиваемся.
6. **Procedure (function) (Процедура или функция)** - это определенная последовательность операторов, то есть набор команд, последовательное выполнение которых приводит к какому-то результату.
7. **Variable (переменная)** - в языках программирования переменные используются для хранения данных определенного типа, например параметров свойств объекта. Каждая переменная имеет свое имя (идентификатор) и хранит только одно значение, которое может меняться в ходе выполнения программы.

ctor=constructor	C++ **конструктор** по умолчанию (не имеющий аргументов, либо возвращающий одно и то же значение для любых значений аргументов)
Mem	**память**, запоминающее устройство (ЗУ)
Member = a component of an object of a given class	**элемент,** в программировании - элемент структуры, класса, объединения или перечисления.
copy-ctor = copy constructor	**копирующий конструктор** в C++ - специальный тип конструктора, вызываемый при копировании объекта
dtor=destructor	**деструктор** - специальный метод, обеспечивающий уничтожение объекта, относящегося к определённому классу, и освобождение занимаемой им памяти. Вся статья >> destructor деструктор (функция-член класса, уничтожающая объект данного класса)
fn=function	1) **функция в программировании** - специальный вид подпрограмм, выполняющих некоторое вычисление и отличающихся тем, что они возвращают в точку вызова результат (return value), присваиваемый имени функции. Поэтому функции, как правило, используются в выражениях. Функция состоит из четырёх частей: типа возвращаемого результата (return type) (он же тип функции), имени функции (function name), списка параметров (parameter list) и тела функции (function body). Функции бывают встроенные (predefined function), библиотечные (library function) и пользовательские (custom function). *The chief advantage of a function is that it can be treated as a "black box" that takes a set of known inputs and produces a corresponding output.* – *Главное преимущество функции в том, что она может рассматриваться как чёрный ящик, принимающий набор известных входных данных и выдающий соответствующий им результат.* 2) **назначение (функция)** например, программы и/или устройства
ptr=pointer =a C/C++ construct declared by: int * p;.	1) **указатель**, 2) **[адресная] ссылка, указатель** а) в программировании - специальный тип данных. Значением переменной или элемента данных этого типа является адрес, который может указывать на другие данные или функцию (прямая адресация), в том числе и на другой указатель (косвенная адресация). Обычно указатели используют при работе с записями, передаче параметров процедурам и организации связанных списков; б) регистр для работы с указателями.

ref=reference *=a C++ construct declared by: int & r;.*	1) **ЯП ссылка** (значение, уникально идентифицирующее объект или переменную во время выполнения; особый вид указателя в C++); 2) **ссылка, адрес, номер [ячейки] в программировании** - элемент данных, значением которого является либо NIL , либо адрес переменной, которая называется ссылочным значением (reference value), или объектом ссылки (referent).
OO=object oriented	1) **объектно-ориентированный, язык программирования**, среда разработки, или приложение, поддерживающие использование объектов; 2) **объектно-ориентированный;** относящийся к объектно-ориентированному подходу (в проектировании или программировании) (отправляющийся не от алгоритмов и программ, а от системы объектов, их свойств и методов).
OOP=object-oriented programming	**объектно-ориентированное программирование (ООП),** доминирующее направление (парадигма программирования, programming paradigm) в развитии программирования; основная идея ООП - представление данных в виде объектов (object), обладающих определёнными свойствами и содержащих внутри себя как структуры данных, так и процедуры для работы с ними (методы). Объекты взаимодействуют друг с другом посредством сообщений (message). ООП имеет развитый аппарат и поддерживается большинством современных языков программирования. Достоинство использования объектной модели в том, что она уменьшает семантический разрыв между предметной областью и программой, а также позволяет писать программы, содержащие на 30% меньше строк исходного текста, что повышает возврат инвестиций. Недостаток - высокая стоимость обучения объектно-ориентированным методам разработки, таким, как UML . Возникновение ООП датируется началом 1970-х годов, когда были созданы языки Simula 67 и Smalltalk.
OOPL= object-oriented programming language	**язык объектно-ориентированного программирования** (напр., C++, Eiffel, Objective C)

method=member function	**метод,** в ООП - **операция, выполняемая объектом**; именованная встроенная процедура или функция, ассоциированная с объектом, которая изменяет состояние объекта или заставляет его отправить сообщение, т. е. реализует поведение объекта (instance behavior) данного класса. Метод может иметь параметры. Существуют также методы класса (class method). Термин был впервые введён в языке SmallTalk . Большинство объектов имеют три типа методов: конструкторы (constructor), деструкторы (destructor) и методы поведения (behavior). Методы также делятся на закрытые (private method) и открытые (public). Syn: member function

Глоссарий C++

Api **Application Programming Interface** an interface made by a system or library to provide access to features and services	**ИПП** **Интерфейс прикладного программирования** набор функций, предоставляемый для использования в прикладных программах
build the process of compiling source code files into executable files	**текущий вариант программы [в процессе разработки]; текущая сборка** результат объединения компонентов программы после внесения в них изменений; компилируется и компонуется из готовых на данный момент версий всех отдельных модулей программы и заглушек; конкретные варианты идентифицируются либо по номеру варианта, либо по дате, а управление процессом автоматизируется при помощи системы управления версиями. Обычно делается ежедневно
cache an area of RAM used to temporarily store data. This reduces the time taken to access the data.	**кэш-память** а) сверхбыстродействующая память (cache memory) для хранения команд и/или данных, к которым наиболее вероятно следующее обращение процессора; б) кэш (быстродействующая буферная память большой ёмкости, используемая для хранения копии областей оперативной памяти с наиболее частым доступом)
executable a file which can be read as by a computer, containing instructions to be executed	**исполнимые (файлы)** любые файлы, которые после загрузки в могут быть непосредственно исполнены - executable file
flush a method of forcing any data in the cache to be written to the output device.	**очистка, сброс** операция опустошения буфера, содержимое которого обычно хранится на другом носителе (в другой памяти); например, содержимое дискового кэша сбрасывается (flushed back) на диск, а процессорного кэша - в основную память

function= FUNC a portion of code which performs a specific task and is relatively independent of the remaining code. Also known as subroutine although this is less often used in C++.	**функция** специальный вид подпрограмм, выполняющих некоторое вычисление и отличающихся тем, что они возвращают в точку вызова результат (return value), присваиваемый имени функции. Поэтому функции, как правило, используются в выражениях. Функция состоит из четырёх частей: типа возвращаемого результата (return type) (он же тип функции), имени функции (function name), списка параметров (parameter list) и тела функции (function body). Функции бывают встроенные (predefined function), библиотечные (library function) и пользовательские (custom function).
header a file which is included into the program, usually for the purposes of reusing the code in other files. Headers typically contain functions.	**заголовок** файл, включенный в программу, обычно с целью повторного использования кода в других файлах. Заголовки обычно содержат функции.
IDE (Integrated Development and Debugging Environment) an integrated development environment, may include a text editor, compiler and debugger or other variations.	**интегрированная среда разработки и отладки [программ]** среда разработки, содержащая редактор исходных текстов программ, систему отладки и другие инструментальные средства, объединённая с компилятором или интерпретатором языка программирования. Позволяет ускорить процесс разработки, создания и отладки программ
integers informally, a "whole number", an integer is any positive natural number (1, 2, 3...), any negative natural number (-1, -2, -3...) or zero (0).	**целые числа** диапазон чисел, которые могут быть представлены в машине и обрабатываются как целые, зависит от её архитектуры и системы команд. Обычно обеспечивается их представление в виде 1-, 2-и 4-байтовых чисел со знаком (signed integer) и без знака (unsigned integer). Например, operations on integers - операции над целыми числами; integer operand - целочисленный операнд, операнд типа integer

runtime error = run-time error an error that occurs upon execution of a program i.e. infinite loop.	**ошибка времени исполнения** программная ошибка, проявляющаяся только при исполнении программы; примеры - логические ошибки (logic error) или выход за границы массива. Подобные ошибки, несмотря на серьёзный контроль во время компиляции и предварительное тестирование, обычно выявляются только при работе в реальной программной среде, с реальными данными. Для обнаружения такого рода ошибок организуется широкое бета-тестирование продукта. Различают контролируемые (checked runtime error) и неконтролируемые (unchecked runtime error) ошибки времени исполнения.
syntax error an error which causes the compiler to fail while creating an executable. The error must be fixed for the code to be compiled successfully.	**синтаксическая ошибка** последовательность символов в программе, нарушающая правила синтаксиса данного языка
variable = var a variable assigns a particular instance of an object type a name or label by which the instance can be referred to. Typically a variable is bound to a particular address in computer memory that is automatically assigned to at runtime, with a fixed number of bytes determined by the size of the object type of a variable and any operations performed on the variable effects one or more values stored in that particular memory location.	**переменная** в процедурном программировании и ООП-именованная область памяти данных, которой программно можно присваивать разные значения (variable value), считывать их и модифицировать. Таким образом, содержимое ячеек этой памяти - это текущее значение переменной. Для использования переменной в программе её необходимо (явно или неявно) объявить: присвоить идентификатор (identifier) и задать тип.
Some simple variables include	**Простые переменные включают**
int - this creates a variable that is an integer (32 bits); **long** - this too creates an	**int -** создает целую переменную (32 бит); **long** - создает целую переменную,

integer variable, but it supports 64 bits; **short** - this again creates an integer, but it only supports 16 bits; **float** - this creates a variable that supports decimals; **double** - this creates a variable that supports MORE decimals; **bool** - this creates a boolean variable, that can only be true or false (1 bit); **char** - this allows one character to be entered (i.e. "W") (7 or 8 bit).	поддерживает 64 битов; **short** – создает переменную, которая предоставляет только 16 бит; **float** – создает переменную, которая поддерживает десятичные числа; **double** – создает переменную, которая поддерживает десятичные числа; **bool** – создает логическая переменная (булева переменная), которая может быть истинной либо ложной (1 бит); **char** – позволяет ввести только один символ (например: “W”) (7 или 8 бит).

Lexemes (лексемы языка C++)

Lexeme – лексема, минимальная содержательная (т. е. имеющая значение) единица языка. Одна из фаз трансляции программы, парсинг (parsing), заключается в лексическом анализе её исходного текста, во время которого из него выделяются лексемы языка (ключевые слова, идентификаторы, литералы и т. д.), затем они передаются синтаксическому анализатору.

preprocessor_tokens = hash | double_hash.	1) **случайные данные;** ненужная информация (в памяти); проф. «мусор» 2) **знак «решетка»** (может обозначать номер, фунт, 16-ричное число, диез и др.) **double hashing** – двойное хеширование 3) **знак "#"** обозначает директиву процессора, такую как например: включить (в файл содержимого другого файла), определить. 4) **знак"##"** используется как директива процессора для разделения маркеров или лексем, рассматриваемых в качестве единого объекта.
keyword (key word) = "asm" | "auto" | "break" | "case" | "char" | "class" | "const" | "continue" | "default" | "delete" | "do" | "double" | "else" | "enum" | "extern" | "float" | "for" | "friend" |	**ключевое слово** 1) в языках программирования – **предопределённое слово, имеющее специальный смысл.** По ключевым словам распознаются заранее определённые действия, типы данных, встроенные функции или операции. Например, print в Бейсике или begin в Паскале. Эти слова резервируются для нужд самого языка и, как правило, не могут использоваться программистом в качестве имён

"goto" | "if" | "inline" | "int" | "long" | "new" | " operator" | "private" | "protected" | "public" | "register" | "return" | "short" | "signed" | "sizeof" | "static" | "struct" | "switch" | "template" | "this" | "throw" | "typedef" | "union" | "unsigned" | "virtual" | "void" | "volatile" | "while".	переменных, функций процедур или методов, поэтому они часто именуются "зарезервированными словами" (reserved word); *Keywords are typed in lower-case letters — ключевые слова печатаются строчными буквами* 2) в СУБД – **слово, по которому может осуществляться поиск** каких-либо записей или документов; например, поиск может вестись по ключевым словам "название фирмы" или "поставщик"; 3) **наиболее информативное слово в заголовке** или тексте документа, описывающее его содержание, используется для индексирования документа; 4) **слово,** описывающее содержание Web-страницы для поисковой машины
punctuation = "(" | "{" | "[" | "]" | "}" | ")" | ";" | ":" | "?" | "\"" | "," | "." .	**пунктуация, расстановка знаков препинания**
operators = "!" | "%" | "^" | "&" | "*" | "-" | "+" | "=" | "|" | "~" | "<" | ">" | "/" | "->" | "++" | ".*" | "->*" | "<<" | ">>" | "<=" | ">=" | "==" | "!=" | "||" | "*=" | "/=" | "%=" | "+=" | "-=" | "<<=" | ">>=" | "&=" | "|=" | "::".	**оператор, знак операции в программировании** - символ (знак, последовательность знаков или ключевое слово), используемый в качестве функции при инфиксной или префиксной записи. Другими словами - это символ, объединяющий операнды в выражения; показывающий, какая операция должна быть выполнена над операндами. Простейшими операторами являются знаки арифметических действий, например сложения. Каждый оператор имеет арность (arity) - число операндов, которые ему необходимы. Операторы делятся на унарные, применяемые к одному операнду (unary operator), и бинарные, применяемые к двум операндам (binary operator). Для определения порядка действий в выражении операторам присваивается старшинство (operator precedence). По типу операций операторы делятся на арифметические (arithmetic operator), логические (logical operator, Boolean operator), присваивания (assignment operator), сравнения (relational operator) и условные (conditional operator). Некоторые ЯВУ позволяют определять и переопределять операторы
identifier (ID)::= #identifier_character ~ keyword & *not starting with a "_" or containing "__".*	**идентификатор, признак; метка** 1) **имена, присваиваемые переменным, константам, структурам данных, классам, процедурам, функциям, методам и другим программным объектам.** Некоторые языки программирования требуют объявления

	идентификаторов до их использования в программе (explicit declaration). Обычно идентификатор представляет собой произвольную последовательность алфавитно-цифровых символов и некоторых специальных знаков типа подчёркивания (их набор может различаться), начинающуюся с буквы. С идентификаторами тесно связано понятие их области видимости (scope); **2) логическое имя устройства –** имя, присваиваемое некоторому устройству или классу устройств для того, чтобы прикладное не зависело от особенностей конструкции устройства. Например, логические имена дисков - A., B., C. и т. д.
identifier_character::= "a".."z" | "A".."Z" | "0".."9" | "_".	**буква, литера, иероглиф** **1) элемент шрифта или кодовой таблицы,** представляющий букву, цифру, знак препинания или другой символ, который может быть введён с клавиатуры, напечатан на принтере или выведен на экран ASC; **2) символ,** знак простой встроенный тип данных в некоторых языках программирования; **3) знак, символ** - в компьютерной и телекоммуникационной терминологии - единица информации, приблизительно соответствующая графеме или графемоподобному элементу, либо символ алфавита или слоговой азбуки письменной формы естественного языка. Примеры - буква, литера, иероглиф, цифра или знак препинания, все символы ASCII и расширенного ASCII , включая пробелы и управляющие знаки. В символьном любые элементы, в том числе графические, появляющиеся на экране, считаются символами. А в графических приложениях этот термин обычно обозначает буквы, цифры и знаки препинания.

Meanings of keywords (значение ключевых слов)

Ключевые слова – это предварительно определенные зарезервированные идентификаторы, имеющие специальные значения. Их использование в программе в качестве идентификаторов не допускается. Для Microsoft C++ зарезервированы следующие ключевые слова. Имена с ведущими символами подчеркивания - расширения microsoft.

Keywords (ключевые слова)	Meaning (значение ключевых слов)
_asm	Ключевое слово **__asm** вызывает встроенный ассемблер и может отображаться везде, где допустим оператор C или C++. Он не может отображаться самостоятельно. За ним должна следовать инструкция по сборке, группа инструкций, заключенная в круглые скобки, либо, в крайнем случае, пустая пара круглых скобок. Термин "блок **__asm**" относится к любой инструкции или группе инструкций, в скобках или без них.При использовании без круглых скобок ключевое слово **__asm** означает, что остальная часть строки- это оператор на языке сборки. При использовании с фигурными скобками оно означает, что каждая строка между скобками — это оператор на языке сборки. Для обеспечения совместимости с предыдущими версиями **_asm** является синонимом **__asm.**
auto	Является описателем объявления. Однако стандарт языка C++ определяет первоначальное и измененное значение данного ключевого слова. До версии Visual C++ 2010 ключевое слово **auto** объявляло переменную в *автоматическом* классе хранения; то есть переменную с локальным временем существования. Начиная с Visual C++ 2010 ключевое слово **auto** объявляет переменную, тип которой выводится из выражения инициализации в соответствующем объявлении. Параметр компилятора /Zc:auto[-] контролирует значение ключевого слова **auto.**
bool	**Целочисленный тип**, который может иметь одно из двух значений: **true** или **false.** Его размер не определен.
break	**Кратковременный останов, приостановка [программы]** инициируемый пользователем останов программы, дающий ему возможность либо решить, что делать дальше, либо выполнить какие-то действия по отладке программы. Обычно осуществляется с помощью специальной комбинации клавиш (break key) на клавиатуре, например Ctrl-C

case	**Конструкция case** в программировании - одно из предложений оператора множественного выбора (multiple choice construct) case, которое будет выполняться только при истинном значении соответствующего условия. **Case** *indicates the start of case in* ***switch*** *statement.*
char **(character)**	**1) буква, литера, иероглиф -** элемент шрифта или кодовой таблицы, представляющий букву, цифру, знак препинания или другой символ, который может быть введён с клавиатуры, напечатан на принтере или выведен на экран; **2) встроенный тип данных** в языках C , C++ и родственных им для объявления символьных переменных и констант; **3) символьный (тип)** (в C/C++ - тип для представления литералов и целочисленных данных); 4) **целочисленный тип**, обычно содержащий члены кодировки выполнения (в Microsoft C++ это кодировка ASCII).
class **(class type)**	Ключевое слово **class** объявляет тип класса или определяет объект типа класса. **Класс, тип объекта** одно из основных понятий. Класс - это объявление структуры данных, объединяющее объекты с одинаковыми свойствами (полями) и методами (поведением). Такие объекты называются экземплярами этого класса и обычно соответствуют сущностям реального мира в деловой или предметной области (то есть это могут быть, например, люди, должности или предметы). Иначе можно сказать, что из определения класса (class definition) в программе можно создать любое количество объектов. Класс иногда называют типом объектов, так как идея объединения в объекте функций с данными тесно связана с понятием типа данных (data type) в процедурных . Класс может наследовать свойства других классов. Свойства объектов могут быть трёх видов: атрибуты (см. attribute), или поля, процедуры или услуги, предоставляемые объектом (см. method) и правила (инварианты), устанавливающие взаимосвязи свойств объекта или определяющие условия его жизнеспособности. *When you need to use the class, you must create an instance of the object (W. Rubin). – Чтобы применить данный класс, необходимо создать экземпляр объекта (invariant).* ***См.*** *abstract class, abstract data type, anonymous class, base class, class diagram, class file, class hierarchy, class invariant, classless inheritance, class interface, class library, class loader, class member, class method, class-oriented, class structure, class variable, concrete class, derived class, final class, friend, metaclass, object type, parameterized class, subclass, superclass, template class.*

command-line switch = command line switch	**Переключатель командной строки** специальный тип аргументов, которые могут быть заданы в командной строке вызова программы для задания нужных режимов её работы, например форматов вывода результатов на экран.
const = constant	1) **Константа, постоянная (величина)**; коэффициент; 2) Для C++ - это **спецификатор константного объекта или неизменяющегося параметра**
const_cast	**Удаляет атрибуты** const, volatile и __unaligned из класса. Указывает на любой тип объекта или элемент данных можно явно преобразовать в идентичный тип, за исключением квалификаторов **const, volatile и unaligned.**
continue	**продолжать, продолжить**, например, выполнение программы с точки останова
decltype	Описывает тип заданного выражения. **decltype** вместе с ключевым словом "auto", используется главным образом для разработчиков, которые создают библиотеки шаблонов.
default	**установленный, используемы по умолчанию, значение по умолчанию** а) значение параметра, состояние, условие, положение переключателей и т. п., предполагаемое при включении, сбросе или инициализации устройства, а также при запуске программы на исполнение. В ходе работы или в процессе конфигурирования оно может быть изменено пользователем; б) заданный заранее вариант ответа, автоматически устанавливаемый программой в диалоговом окне, если пользователь не сделал другого выбора. ***см.*** *default action, default configuration, default directory, default drive, default option, default parameter, default route, default user, default value*
delete (del)	Отменяет выделение блока памяти. [::] delete cast-expression [::] delete [ ] cast-expression
do (digital output) **.DO**	**1) выход цифровых данных 2) цифровые выходные данные** **(расширение файла)** команды управления процессом размещения компонентов и трассировки проводников (пакет ACCEL EDA)
double	Это тип с плавающей запятой, размер которого больше или равен размеру типа float, но меньше или равен размеру типа long double; **long double** – это тип с плавающей запятой, размер которого равен размеру типа double.
dynamic_cast	Преобразует операнд *expression* в объект типа *type-id*.

else	**Иначе; Управляет условным ветвлением.**
enum	Это тип перечисление. В enum записываются константы и каждой новой константе присваивается порядковый номер
__event	Ключевое слово может быть применено к объявлению метода, объявлению интерфейса или объявлению данных-членов. Однако ключевое слово **__event** не может использоваться для уточнения члена вложенного класса.
explicit	Ключевое слово **explicit** сообщает компилятору, что указанное преобразование нельзя использовать для выполнения неявных преобразований. Если вы хотели воспользоваться преимуществами синтаксиса неявных преобразований до введения ключевого слова **explicit**, вам приходилось либо смиряться с нежелательными последствиями, иногда возникающими в связи с применением неявного преобразования, или использовать в качестве обходного пути менее удобную функцию именованного преобразования. Теперь же, используя ключевое слово **explicit** можно создавать удобные преобразования, которые можно использовать только для выполнения явного приведения или прямой инициализации, не опасаясь возникновения проблем, подобных проблеме safe bool.
extern	**внешний** (спецификация объекта или функции, которым придается внешнее связывание (external linkage))
false	Это ключевое слово является одним из двух значений переменной типа bool или условного выражения (условное выражение теперь является **истинным**логическим выражением). Например, если i является переменной типа **bool**, оператор i = false; присваивает значение **false** переменной i.
float	**Э**то тип с плавающей запятой наименьшего размера.
for	Обозначает начало цикла с параметром. **for(i=0; i<10; i++) ...** **for(i=10; i>=0; i--) ...** **for(i=0,j=10; i<10; i++,j--) ...** ***for(A;B;C)D* means the same as *{A; while(B){{D} C; }}***
friend	**Дружественный,** в C++ - класс или операция (метод объекта), имеющие привилегированный доступ к закрытой (private) части (реализации) другого объекта, на которые "недружественные" классы и операции ссылаться не могут.
_hook	Связывает метод обработчика с событием.

if	**Оператор if** служит для того, чтобы выполнить какую-либо операцию в том случае, когда условие является верным. *Условная конструкция в C++* всегда записывается в круглых скобках после оператора if.
	Внутри фигурных скобок указывается тело условия. Если условие выполнится, то начнется выполнение всех команд, которые находятся между фигурными скобками.
	Пример конструкции ветвления
	if (num < 10) { // Если введенное число меньше 10.
	cout << "Это число меньше 10." << endl;
	} **else** { // иначе
	cout << "Это число больше либо равно 10." << endl;
	}
	Здесь говорится: «**Если** переменная **num** меньше 10 — вывести соответствующее сообщение. **Иначе**, вывести другое сообщение».

__if_not_exists	Оператор проверяет, существует ли указанный идентификатор. Если идентификатор не существует, выполняется определенный блок операторов. Данный оператор __if_not_exists применяют только к простым типам, а не шаблонам; к идентификаторам как внутри, так и вне класса. Не применяют оператор __if_not_exists к локальным переменным. Его используют только в теле функции. За пределами тела функции оператор __if_not_exists может проверять только полностью определенные типы. При проверке перегруженных функций невозможно выполнить проверку определенной формы перегрузки. Дополнением к оператору __if_not_exists является оператор __if_exists.
inline	Ключевое слово inline доступно только в C++. Ключевые слова __inline и __forceinline доступны как в C, так и в C++. Для обеспечения совместимости с предыдущими версиями ключевые слова _inline и __inline являются синонимами. Ключевое слово **inline** сообщает компилятору, что подстановка является предпочтительной. Однако компилятор может создать отдельный экземпляр функции и вместо подстановки кода создать стандартную компоновку вызова. Существует две ситуации, в которых это может происходить.
literal	Переменная (элемент данных), помеченная как literal в компиляции /clr является эквивалентом static const переменной.

mutable	Это ключевое слово может применяться только к данным-членам класса, которые не являются статическими или константами. Если данные-члены объявлены как **mutable**, им можно присваивать значения из функции-члена со спецификатором **const**.
inline	1) встраиваемый (напр. о подпрограмме) 2) встроенный (напр. об изображении внутри текста)
int	1) **целочисленный тип**, размер которого больше или равен размеру типа **short int** и меньше или равен размеру типа **long**. 2) сигнал INT обозначение входа сигнала прерывания (INT signal) и/или самого этого сигнала у некоторых микропроцессоров; 3) interrupt (расширение файла) файлы с текстом интерфейсных частей модулей в Турбо-Паскале
long **(или long int)**	**Целочисленный тип**, который больше или равен размеру типа int. Объекты типа **long** могут объявляться как объекты типа **signed long** и **unsigned long**. **Signed long** – синоним **long**.
longlong	Больше, чем беззнаковый **long**. Объекты типа **long long** могут объявляться как объекты типа **signed long long** и **unsigned long long**. **Signed long long** = **long long**.
naked	Блок, относящийся только к системам Microsoft. Код функций, объявленных с атрибутом **naked**, создается компилятором без кода пролога и эпилога. Эту функцию можно использовать, чтобы написать собственные коды пролога и эпилога, используя встроенный ассемблерный код. Функции с атрибутом naked особенно полезны для написания драйверов виртуальных устройств. Обратите внимание, что атрибут **naked** допускается только для архитектур x86 и ARM; на платформе x64 он недоступен.
namespace	Объявление **пространства имен** определяет и присваивает уникальное имя используемого объявленного пространства имен.Такие пространства имен используются для устранения проблемы конфликтов имен в больших программах и библиотеках. Программисты могут использовать пространства имен для разработки новых программных компонентов и библиотек, не вызывая конфликтов имен с существующими компонентами.
new	1) Ключевое слово **new** указывает, что виртуальный член получит новую ячейку в таблице vtable. 2) оператор **new** выделяет память для объекта или массива объектов типа *type-name* из свободного хранилища и возвращает подходяще типизированный, не нулевой указатель на объект.

operator	Данное ключевое слово предоставляет объявление функции, указывающей что operator-symbol означает при его применении к экземплярам класса. Это предоставляет оператору многозначность или "перегружает" его. Компилятор различает разные виды одного и того же оператора, изучая типы его операндов.
public	Если ключевое слово **public** располагается перед списком членов класса, оно указывает, что эти члены доступны из любой функции. Это применяется ко всем членам, объявленным до следующего описателя доступа или до конца класса. Если ключевое слово **public** располагается перед именем базового класса, оно указывает, что открытые и защищенные члены базового класса являются, соответственно, открытыми и защищенными членами производного класса.
private	Если ключевое слово **private** располагается перед списком членов класса, оно указывает, что эти члены доступны только в функциях-членах и дружественных объектах класса. Это применяется ко всем членам, объявленным до следующего описателя доступа или до конца класса. Если ключевое слово **private** располагается перед именем базового класса, оно указывает, что открытые и защищенные члены базового класса являются закрытыми членами производного класса.
protected	Ключевое слово **protected** определяет доступ к членам класса в *списке-членов* до следующего описателя доступа (**public** или **private**) или окончания определения класса. Члены класса, объявленные как **protected**, могут использоваться только следующими объектами. • Функции-члены класса, в котором впервые были объявлены эти элементы. • Дружественные элементы класса, который изначально объявил эти элементы. • Классы, производные с использованием открытого или защищенного доступа от класса, который изначально объявил эти элементы. • Прямые производные с использованием закрытого доступа классы, которые также имеют закрытый доступ к защищенным элементам. Если ключевое слово **protected** располагается перед именем базового класса, оно указывает, что открытые и защищенные члены базового класса являются защищенными элементами соответствующих производных классов.

ref new	Агрегатное ключевое слово **ref new** выделяет экземпляр типа, который уничтожается сборщиком мусора, когда объект становится недоступным, и возвращает дескриптор (^) для выделенного объекта.
register	Ключевое слово **register** указывает, что переменную по возможности нужно сохранить в регистре компьютера.
reinterpret_cast	Позволяет преобразовывать любой указатель в указатель любого другого типа. Также позволяет преобразовывать любой целочисленный тип в любой тип указателя и наоборот. Неправильное использование оператора **reinterpret_cast** легко может оказаться небезопасным. Если требуемое преобразование не является низкоуровневым по самой своей природе, следует использовать один из других операторов приведения типов. Оператор **reinterpret_cast** можно использовать для таких преобразований, как char* в int* или One_class* в Unrelated_class*, которые небезопасны по самой своей сути. Результат применения **reinterpret_cast** не может безопасно использоваться ни для чего другого, кроме приведения обратно в исходный тип. Другие применения в лучшем случае будут непереносимыми.
return	Завершает выполнение функции и возвращает элемент управления в вызывающую функцию (или в операционную систему при передаче управления из функции **main**). Выполнение возобновляется в вызывающей функции в точке сразу после вызова.
safe_cast	Операция **safe_cast** в случае успешного выполнения возвращает указанное выражение как указанный тип; в противном случае возникает исключение **Invalid Cast Exception**.
short int (или просто short)	Это целочисленный тип, размер которого больше или равен размеру типа **char** и меньше или равен размеру типа **int**.
sizeof	Возвращает размер своего операнда относительно размера типа **char**.Оператор **sizeof** может иметь один из следующих операндов: • Имя типа. Если оператор **sizeof** используется с именем типа, оно должно быть заключено в скобки. • Выражения. Если оператор **sizeof** используется с выражением, его можно определять как со скобками, так и без них. Значение выражения не вычисляется.

static	Ключевое слово **static** можно использовать для объявления переменных, функций, данных-членов класса и функций класса. По умолчанию объект или переменная, определенные вне всех блоков, имеют статическую длительность и внешнюю компоновку. Статическая длительность означает, что объект/переменная размещается в памяти при запуске программы, а освобождает память, когда программа завершается. Внешняя компоновка означает, что имя переменной видно за пределами файла, в котором эта переменная объявлена. Внутренняя компоновка означает, что имя не видно за пределами файла, в котором объявлена переменная.
struct	Ключевое слово **struct** указывает тип структуры или переменную типа структуры.
switch	Позволяет осуществить выбор среди нескольких фрагментов кода, в зависимости от значения целочисленного выражения, осуществляет **запуск оператора выбора** (switch statement) **switch(i)** **{** **case 2: i=3; break;** **case 3: i=2; break;** **}** **Switch** *is a start of a switch statement.*
switch statement	**оператор выбора (ветвления)**
this	Указатель **this** доступен только в нестатических функциях-членах типа **class**, **struct** или **union**. Он указывает на объект, для которого вызывается функция-член. Статические функции-члены не имеют указатель **this**.
throw	Для реализации обработки исключений в C++ используют выражения throw, try, и catch. Сначала используют блок try для включения одного или нескольких операторов, которые могут создавать исключение. Выражение throw означает, что исключительное условие, что часто является ошибкой, произошло в блоке try. В качестве операнда выражения throw можно использовать объект любого типа. Обычно этот объект используется для передачи информации об ошибке. В большинстве случаев рекомендуется использовать класс std::exception или один из производных классов, определенных в стандартной библиотеке. Если один из них не подходит, рекомендуется создать собственный производный класс исключений из std::exception. Для обработки исключений, которые могут быть созданы, необходимо реализовать один или несколько блоков **catch** сразу после блока try. Каждый блок catch указывает тип исключения, которое он может обрабатывать.

true	Это ключевое слово является **одним из двух значений переменной типа bool** или условного выражения (условное выражение теперь является истинным логическим выражением). Если i принадлежит типу **bool**, оператор i = true; присваивает i значение **true**.
try-finally	Оператор является **расширением Microsoft для языков C и C++** и позволяет целевым приложениям гарантировать выполнение кода очистки при прерывании выполнения блока кода. Очистка включает такие задачи, как отмена распределения памяти, закрытие файлов и освобождение их дескрипторов. Оператор **try-finally** особенно полезен для подпрограмм, в которых в нескольких местах выполняется проверка на наличие ошибок, способных вызвать преждевременное возвращение из подпрограммы.
typedef	Объявление **typedef**, которое содержит имя, которое внутри своей области является синонимом для типа, указанного частью *объявления type-declaration*. Объявления typedef можно использовать для создания более коротких и значимых имен для типов, определенных в языке, или для типов, которые объявили вы. Имена typedef позволяют инкапсулировать детали реализации, которые могут измениться. В отличие от объявлений **class**, **struct**, **union** и **enum**, объявления **typedef** не вводит новый тип — они вводят новые имена для уже существующих. Имена typedef разделяют пространство имен с обычными идентификаторами. Поэтому программа может иметь имя typedef и идентификатор локальной области с одним и тем же именем.
union	Это пользовательский тип данных или класса, который в любой конкретный момент содержит только один объект из списка своих членов (при этом такой объект может представлять собой тип массива или класса).
virtual	Ключевое слово языка C++, объявляющее **виртуальную функцию - virtual function**. If a member function f of class B is defined to be virtual and a B* variable v is pointing at an object of derived class D then a call **v->f(whatever)** will call the version of f defined in D (if any) rather than B. In short this use of virtual indicates dynamic dispatching. If class D is derived from two classes X and Y and both X and Y are derived from B, then the word virtual indicates that only one "copy" of B will be present in D.

void	Ключевое слово в языках C++ , Java и ряде других, стоящее перед определением функции и означающее, что она не возвращает никакого значения, т. е. возвращаемое значение пусто или не существует. *The main method is declared void because it doesn't return a value. - Метод main объявлен как void, потому что он не возвращает значение.* Если ключевое слово **void** указывает возвращаемый тип функции, оно означает, что данная функция не возвращает никакого значения. Если оно используется для списка параметров функции, оно означает, что функция не принимает никаких параметров. Если оно используется в объявлении указателя, оно означает, что указатель является "универсальным". Если указатель имеет тип **void ***, он может указывать на любую переменную, объявленную без указания ключевого слова **const** или **volatile**. Указатель с ключевым словом **void** не может быть разыменован, кроме как путем приведения к другому типу. Указатель с ключевым словом **void** может быть преобразован в любой другой тип указателя на данные. В C++ указатель с ключевым словом **void** может указывать на функцию, но не на член класса. Объявить переменную типа **void** невозможно.
volatile	Квалификатор, который используется для объявления о том, что объект может быть изменен в программе аппаратным обеспечением.
while	Циклически выполняет *оператор* до тех пор, пока значение параметра *выражение* не будет равно нулю.
__wchar_t	Переменная **__wchar_t** обозначает тип расширенных символов или многобайтовых символов. По умолчанию тип wchar_t является собственным типом, но можно использовать /Zc: wchar_t-, чтобы сделать wchar_t определением типа для **unsigned short.**

Основные типы языка C++

Категория	Type	Описание
Целые числа	**char**	**char** - это целочисленный тип, обычно содержащий члены кодировки выполнения (в Microsoft C++ это кодировка ASCII).
	char, **signed char, unsigned char**	Компилятор C++ обрабатывает переменные типа **char**, **signed char** и **unsigned char** как переменные разных типов. Переменные типа char повышается до int , если они имеют тип signed char по умолчанию, и не используется параметр компиляции /J. В этом случае они рассматриваются как тип **unsigned char** и повышаются до типа **int** без расширения знака.
	bool	**bool** - это целочисленный тип, который может иметь одно из двух значений: **true** или **false**. Его размер не определен.
	short	**short int** (или просто **short**) - это целочисленный тип, размер которого больше или равен размеру типа **char** и меньше или равен размеру типа **int**.
	short	Объекты типа **short** могут объявляться как объекты типа **signed short** и **unsigned short**. **Signed short** - синоним **short**.
	int	**int** — это целочисленный тип, размер которого больше или равен размеру типа **short int** и меньше или равен размеру типа **long**. Объекты типа int могут быть объявлены как **unsigned int** или **signed int**. **Signed int** - синоним **int**.
	__int*n*	Целое значение заданной размерности, где *n* - размер целого числа в битах. Значение *n* может быть 8, 16, 32 или 64 (**__int***n* - ключевое слово для систем Microsoft).
	long	Тип **long** (или **long** int) - это целочисленный тип, который больше или равен размеру типа int.

	signed long	Объекты типа **long** могут объявляться как объекты типа **signed long** и **unsigned long**. **Signed long** - синоним **long**.
	longlong	Больше, чем беззнаковый **long**. Объекты типа **long long** могут объявляться как объекты типа **signed long long** и **unsigned long long**. **Signed long long** - синоним **long long**.
Числа с плавающей запятой	**float**	**float** - это тип с плавающей запятой наименьшего размера.
	double	**double** - это тип с плавающей запятой, размер которого больше или равен размеру типа **float**, но меньше или равен размеру типа **long double**.
	long double 1	**long double** - это тип с плавающей запятой, размер которого равен размеру типа **double**.
Расширенные символы	**_wchar_t**	Переменная **__wchar_t** обозначает тип расширенных символов или многобайтовых символов. По умолчанию тип **wchar_t**является собственным типом, но можно использовать /Zc: wchar_t-, чтобы сделать **wchar_t** определением типа для **unsigned short**. Чтобы указать константу расширенного символьного типа, перед символьной или строковой константой следует использовать префикс L.

bit Binary Digit. That is, a variable quantity capable of holding exactly one of two possible values. By convention, these values are labeled 0 (zero) and 1 (one). A **byte** is 8 bits. A **kilobyte** (KB) is 2^{10} bytes = 1024 bytes. A **megabyte** (MB) is 2^{20} bytes = 1024KB. A **gigabyte** (GB) is 2^{30} bytes = 1024 MB. A **terabyte** (TB) is 2^{40} bytes = 1024 GB. A **petabyte** (PB) is 2^{50} bytes = 1024 TB. An **exabyte** (EB) is 2^{60} bytes = 1024 PB.	**бит** Двоичная цифра, то есть, переменная величина, способная принимать только одно из двух возможных значений 0 (ноль) или 1 (один). **1 байт** – 8 **бит.** **1 килобайт** (КВ) – 2^{10} байтов = 1024 байт. **1 мегабайт** (МБ) – 2^{20} байтов = 1024 КВ. **1 гигабайт** (GB) – 2^{30} байтов = 1024 МБ. **1 терабайт** (ТВ) – 2^{40}байтов = 1024 GB. **1 петабайт** (РВ) – 2^{50} байтов = 1024 ТВ. **1 эксабайт** (ЕВ) – 2^{60} байтов = 1024 РВ.

Логический тип
bool - тип, способный хранить одно из двух значений: true (истина) или false (ложь).

Символьные типы (в C/C++ - тип для представления литералов и целочисленных данных)
signed char - тип для знакового представления символов.
unsigned char - тип для беззнакового представления символов.
char - тип для представления символов, который может наиболее эффективно обрабатываться в целевой системе
(эквивалентный signed char или unsigned char, но всё же отличный от них тип).
wchar_t - тип для широкого представления символов.
char16_t - тип для представления символов в UTF-16. (начиная с C++11)
char32_t - тип для представления символов в UTF-32. (начиная с C++11)

Целочисленные типы
int — базовый целочисленный тип. Может быть опущен, если представлен любой из модификаторов. Если не представлен ни один из модификаторов размера, гарантировано имеет ширину не меньше 16 бит. Тем не менее, на 32/64-битных системах почти всегда имеет ширину не меньше 32 бит.

Модификаторы
Модифицируют целочисленный тип. Могут располагаться в любом порядке. Только один модификатор из каждой группы может быть представлен в имени типа.

Знаковость
signed — целевой тип будет иметь знаковое представление (по умолчанию, если не представлен ни один из вариантов).
unsigned — целевой тип будет иметь беззнаковое представление.

Размер
short — целевой тип будет оптимизирован по размеру и иметь ширину не меньше 16 бит.
long — целевой тип будет иметь ширину не меньше 32 бит.
long long — целевой тип будет иметь ширину не меньше 64 бит.

Описание алгоритмов (Обозначения)
1. **first, last** - итераторы (управляющая структура, определяющая порядок выполнения некоторых повторяющихся действий, устройство или программа организации циклов) конца и начала (first1, last1, first2, last2 — итераторы конца и начала 1 и 2 диапазона соответственно)
2. **middle** - итератор, указывающий на определенную позицию в контейнере
3. **function, predicate, op и comp** - функциональные объекты
4. **value, new, old и init** - значения объектов, хранящихся в контейнерах
5. **a, b** - некоторые объекты одного типа
6. **iter –** итератор

Неизменяющиеся последовательные операции

Название функции	Аргументы функции	Описание функции
adjacent_find	first, last	Возвращает итератор, указывающий на первую пару одинаковых объектов
count	first, last, value	Возвращает количество элементов, значение которых равно value
equal	first1, last1, first2	Возвращает значение true, если все соответствующие пары объектов из двух диапазонов равны
find	first, last, value	Возвращает итератор, указывающий на первый элемент, равный значению value
for_each	first, last, function	Применяет function ко всем объектам
mismatch	first1, last1, first2	Возвращает первую несовпадающую пару соответствующих объектов, расположенных в разных диапазонах позиций контейнера
search	first1, last1, first2, last2	Проверяет, содержится ли второй диапазон внутри первого, возвращает начало совпадения или last1, если нет совпадения

Изменяющиеся последовательные операции

Название функции	Аргументы функции	Описание функции
fill	first, last, value	Присваивает значение value всем объектам из диапазона
generate	first, last, gen	Заполняет диапазон значениями, получаемыми с помощью последовательных вызовов функции gen
iter_swap	iter1, iter2	Обменивает объекты, на которые указывают два итератора
remove	first, last, value	Удаляет из диапазона все значения, равные value
reverse	first, last	Обращает последовательность объектов из диапазона
replace	first, last, old, new	Заменяет все объекты, равные old , объектами, равными new
rotate	first, last, middle	Отражает зеркально последовательность элементов
swap	a, b	Заменяет один объект другим

swap_ranges	first1, last1, first2	Обменивает соответствующие объекты в двух диапазонах
transform	first1, last1, first2, operator	Превращает объекты из диапазона 1 в новые объекты диапазона 2, применяя operator
unique	first, last	Удаляет все эквивалентные объекты в последовательности, кроме первого

Операции сортировки

Название функции	**Аргументы функции**	**Описание функции**
nth_element	first, nth,last	Помещает n-й объект в позицию, которую он занимал бы после сортировки всего диапазона
sort	first, last	Сортирует объекты в диапазоне
stable_sort	first, last	Сортирует объекты в диапазоне. Если два объекта равны, их порядок не изменится.

Бинарные операции поиска

Название функции	**Аргументы функции**	**Описание функции**
binary_search	first, last, value	Возвращает true, если значение value входит в интервал
equal_range	first, last, value	Возвращает пару объектов, представляющих собой нижнюю и верхнюю границы, между которыми можно вставить значение value без изменения порядка сортировки
lower_bound	first, last, value	Возвращает итератор, указывающий на первую позицию, в которую можно вставить значение value без изменения порядка следования объектов
upper_bound	first, last, value	Возвращает итератор, указывающий на последнюю позицию, в которую можно вставить значение value без изменения порядка следования объектов

Операции слияния

Название функции	Аргументы функции	Описание функции
includes	first1, last1, first2, last2	Возвращает true, если все объекты из диапазона first2 last2 имеются также в диапазоне first1 (только для работы с set и multiset)
merge	first1, last1, first2, last2, first3	соединяет отсортированные диапазоны 1 и 2 в диапазон 3
set_difference	first1, last1, first2, last2, first 3	Создает упорядоченную разность множеств, заданных диапазонами 1 и 2 (только для работы с set и multiset)
set_intersection	first1, last1, first2, last2, first3	Создает упорядоченное пересечение элементов диапазонов 1 и 2 (только для работы с set и multiset)
set_union	first1, last1, first2, last2, first3	Создает упорядоченное объединение элементов диапазонов 1 и 2 (только для работы с set и multiset)

Операции отношений

Название функции	Аргументы функции	Описание функции
lexicographical_compare	first1, last1, first2, last2	Возвращает true, если последовательность в диапазоне 2 следует в алфавитном порядке за последовательностью диапазона 1
max	a, b	Возвращает наибольшее из a, b
max_element	first, last	Возвращает итератор, указывающий на наибольший объект в диапазоне
min	a, b	Возвращает наименьшее из a, b
min_element	first,last	Возвращает итератор, указывающий на наименьший объект в диапазоне
next_permutation	first, last	Выполняет одну перестановку в последовательности данного диапазона
prev_permutation	first, last	Выполняет одну обратную перестановку в последовательности данного диапазона

Кучи

Название функции	Аргументы функции	Описание функции
make_heap	first, last	Создает кучу из значений диапазона first last
pop_heap	first, last	Меняет значения в first и last-1. Помещает диапазон first last-1 в кучу
push_heap	first, last	Помещает значение из last-1 в результирующую кучу (heap, область динамической памяти) диапазон от first до last
sort_heap	first, last	Упорядочивает элементы в куче first last

Контейнеры

<bitset> Реализует специализированный класс контейнеров std::bitset - битовый массив.

<deque>Реализует шаблон класса контейнера std::deque - двусвязная очередь.

<list>Реализует шаблон класса контейнера std::list - двусвязный список.

<map>Реализует шаблоны классов контейнеров std::map и std::multimap - Ассоциативный массив и мультиотображение.

<queue> Реализует класс адаптер-контейнера std::queue - односторонняя очередь.

<set> Реализует шаблоны классов контейнеров std::set и std::multiset - сортированные ассоциативные контейнеры или множества.

<stack> Реализует класс адаптер-контейнера std::stack - стек.

<vector> Реализует шаблон класса контейнеров std::vector - динамический массив.

Общие

<algorithm> Реализует определения многих алгоритмов для работы с контейнерами.

<functional> Реализует несколько объект-функций, разработанных для работы со стандартными алгоритмами.

<iterator>Реализует классы и шаблоны для работы с итераторами.

<locale>Реализует классы и шаблоны для работы с локалями.

<memory>Реализует инструменты управления памятью в C++, включая шаблон класса std::auto_ptr.

<stdexcept> Содержит стандартную обработку ошибок классов, например, std::logic_error и std::runtime_error, причем оба происходят изstd::exception.

<utility> реализует шаблон класса std::pair для работы с парами (двучленными кортежами) объектов.

Строковые

<string>Реализует стандартные строковые классы и шаблоны.

Поточные и ввода-вывода

<fstream>Реализует инструменты для файлового ввода и вывода. Смотри fstream.
<ios>Реализует несколько типов и функций, составляющих основу операций с iostreams.
<iostream>Реализует основы ввода и вывода языка C++. Смотри iostream.
<iosfwd>Реализует предварительные объявления нескольких шаблонов классов, связанных с вводом-выводом.
<iomanip>Реализует инструменты для работы с форматированием вывода, например базу, используемую при форматировании целых и точных значений чисел с плавающей запятой.
<istream>Реализует шаблон класса std::istream и других необходимых классов для ввода.
<ostream>Реализует шаблон класса std::ostream и других необходимых классов для вывода.
<sstream>Реализует шаблон класса std::sstream и других необходимых классов для работы со строками.

Числовые

<complex> Реализует шаблон класса std::complex и связанные функции для работы с комплексными числами.
<numeric> Реализует алгоритмы для числовой обработки.
<valarray> Реализует шаблон класса std::valarray - класс массивов, оптимизированный для числовой обработки.

Языковая поддержка

<exception> Реализует несколько типов и функций, связанных с обработкой исключений, включая std::exception — базовый класс всех перехватов исключений в Стандартной Библиотеке.
<limits> реализует шаблон класса std::numeric_limits, используемый для описания свойств базовых числовых типов.
<new> Реализует операторы new и delete, а также другие функции и типы, составляющие основу управления памятью в C++.
<typeinfo>Реализует инструменты для работы с динамической идентификацией типа данных в C++.

REFERENCE SOURCES

1. [Электронный ресурс]: Режим доступа: http://mex.onu.edu.ua/studying/tasks1/
2. [Электронный ресурс]: Режим доступа: http://www.helloworld.ru/texts/comp/lang/c/c14/index.html
3. [Электронный ресурс]: Режим доступа: http://nullpro.info/2013
4. [Электронный ресурс]: Режим доступа: http://www.codenet.ru/progr/visualc/vc/
5. [Электронный ресурс]: Режим доступа: http://www.opita.net/doc/cpp