**Libstdc++**

**Include**

**/Backward:**

**Auto\_ptr.h:**

Файл содержит только реализацию класса auto\_ptr. Включает в себя конструкторы, деструктор, переопределенные операторы присваивания и разыменования.

**Backward\_warning.h:**

файл-заглушка, не содержит исполняемого кода.

**Binders.h:**

Файл, содержащий реализацию классов binder1st и binder2nd. Они предоставляют возможность связать аргументы (первый и второй соответственно) с аргументами двоичной функции. Содержат конструкторы и перепределенный оператор ().

**Hash\_fun.h:**

Файл, содержащий реализацию структур hash для различных типов данных. В самих структурах содержится только переопределенный оператор (). Для строк выполняется хеширование, а для всех остальных типов – возвращается переданное в качестве аргумента значение, при это ничего не вызывается.

**Hash\_map.h:**

Файл, содержащий реализацию класса hash\_map, являющимся старым аналогом std::unordered\_map из библиотеки STL.

**Hash\_set.h:**

Файл, содержащий реализацию класса hash\_set, являющимся старым аналогом std::unordered\_set из библиотеки STL.

**Hashtable.h:**

Файл, содержащий реализацию класса hashtable, используемый для реализации ассоциативных контейнеров типа hash\_set, hash\_map, hash\_multiset, hash\_multimap.

**Strstream.h:**

Файл содержит 4 класса :

Strstreambuf, istrstream, ostrstream, strstream.

Strstreambuf:

Класс представляет собой интерфейс для работы со статическим массивом символов. Устаревший аналог класса stringbuf;

Istrstream:

Реализует операции ввода для потоков с поддержкой массива. Является оберткой более выского уровния над std::basic\_istream. Также это устаревший аналог класса istringstream;

Ostrstream:

Реализует операции вывода для потоков с поддержкой массива. Является устаревшим аналогом класса ostringstream;

Strstream:

Класс реализует операции ввода и вывода для потоков с поддержкой массива. Является устаревшим аналогом класса stringstream;

/**Bits:**

**Atomic\_base.h:**

Файл, содержащий реализацию атомарных вычислений (инкремент, декремент, сложение, вычитание и пр.).

**Basic\_ios.h:**

Файл, содержащий реализацию класса Std::basic\_ios, который предоставляет средства для взаимодействия с объектами, которые имеют интерфейс std::basic\_streambuf. Содержит конструктор, деструктор и следующие методы:

Good – функция, проверяющая не возникло ли ошибок в процессе ввода-вывода и доступны ли эти функции

Eof – функция, проверяющая достигнут ли конец файла;

Fail- функция, проверяющая наличие ошибки;

Bad – функция, проверяющая наличие критической ошибки;

Operator ! – функция, возвращающая результат fail;

Operator bool – функция, возвращающая результат !fail();

Imbue – устанавливает язык;

Tie – связывает текущий поток с выходным;

**Char\_traits.h:**

Файл, содержащий в себе реализацию класса Std::Char\_traits. Класс char\_traits определяет поток и свойства строковых операций для типа char, а также методы для управления символами, строками и общие операции по работе с типом char.

**Forward\_list.h:**

Файл содержит реализацию класса std::forward\_list из библиотеки STL, а также реализацию его итератора.

**Functexcept.h:**

Файл содержит реализацию вспомогательных функций для генерации исключений различного вида (при аллокации, приведении типов, определенных в классе std::except, и пр.)

**Gslice.h:**

Файл, содержащий реализацию контейнера, представляющего с собой многомерное подмножество массива. Определены следующие методы:

Конструтор, деструктор, оператор присваивания, функцию доступа к первому элементу, функцию получения размера контейнера.

**Indirect\_array.h:**

Файл, содержащий реализацию соответствующего класса, представляющего собой ссылку на произвольное подмножество элементов массива, заданных упорядоченных массивом индексов. В данном классе определены следующие методы:

Переопределенные операторы всех унарных операций, оператор присваивания, конструкторы и деструктор.

**Mask\_array.h:**

Файл, содержащий реализацию класса mask\_array, представляющего собой ссылку на указанное подмножество элементов массива, заданного битовой маской, содержащей булевые значения (0 – элемент исходного массива не включен в mask\_array, 1 – включен). В данном классе есть конструктор, деструктор, оператор присваивания, а также переопределены унарные операторы.

**Move.h:**

Файл, содержащий в себе реализацию функции swap, меняющую переданные

аргументы местами.

**Range\_access.h:**

Файл содержит реализацию функций типа begin, end, rbegin, rend для различных типов контейнеров.

**Regex.h:**

Файл, реализующий интерфейс для создания и обработки регулярных выражений. Содержит конструкторы, операторы присваивания, функцию assign, compare для создания и сравнения регулярных выражений.

**Shared\_ptr.h**

Файл, содержащий в себе реализацию класса shared\_ptr. Включает в себя конструкторы, деструктор, переопределенные операторы сравнения и присваивания, функцию make\_shared для создания объекта shared\_ptr из имеющегося указателя

**Slice\_array.h:**

Файл, содержащий в себе реализацию класса slice, который является одномерным подмножеством массива. Содержит в себе конструкторы, деструктор, оператор присваивания, переопределенные унарные операторы.

**Stl\_algo.h:**

Файл который содержит реализацию функции из библиотеки algorithm.

**Stl\_construct.h:**

Файл, содержащий реализацию функций для создания объектов произвольного типа, выделения для них памяти и их инициализации, а так же функции для их уничтожения.

**Stl\_deque.h:**

Файл, содержащий в себе реализацию класса std::deque из библиотеки STL, а также реализацию его итератора.

**Stl\_function.h:**

Файл, содержащий реализацию класса std::function. В нем в виде функциональных объектов (функторов) реализованы функции сравнения (>, <, >= , <=, и т.п.) а также булевы функции (дизъюнкция, конъюнкция, исключающее ИЛИ и т.п.)

**Stl\_iterator.h:**

Файл содержит реализацию класса std::iterator из библиотеки STL. Также предоставлена реализация функций инкремента, декремента, оператора разыменования и присваивания для различных типов итераторов.

**Stl\_list.h:**

Файл содержит реализацию класса std::list из библиотеки STL. Определены функции итерирования по элементам, конструктор, деструктор, функции удаления и вставки элементов, переопределены операторы сравнения и присваивания.

**Stl\_map.h:**

Файл, содержащий реализацию класса std::map из библиотеки STL. Определены функции вставки и удаления элементов, переопределенные операторы сравнения, поиска, получения begin/end, и функции upper/lower bound .

**Stl\_multimap.h:**

Файл, содержащий реализацию класса std::multimap из библиотеки STL. Определены функции вставки и удаления элементов, переопределенные операторы сравнения, поиска, получения begin/end а также функции upper/lower bound.

**Stl\_multimap.h:**

Файл содержит реализацию класса std::multimap из библиотеки STL. Данный класс представляет собой ассоциативный контейнер, который содержит упорядоченный список пар "ключ-значение".

**Stl\_multiset.h:**

Файл содержит реализацию класса std::multiset из библиотеки STL. Данный класс представляет собой ассоциативный контейнер, который содержит упорядоченный набор объектов типа “Ключ”. В отличие от set, в multiset допускаются ключи с одинаковыми значениями.

**Stl\_numeric.h:**

Файл содержит реализацию класса std::numeric из библиотеки STL. Данный класс предоставляет интерфейс для работы с массивами чисел.

**Stl\_pair.h:**

Файл содержит реализацию класса std::pair из библиотеки STL. Данный класс представляет собой шаблон структуры, который предоставляет возможность хранить два разнородных объекта, как единое целое.

**Stl\_queue.h:**

Файл содержит реализацию класса std::queue из библиотеки STL. Данный класс представляет собой контейнерный адаптер, который дает функциональность очереди - в частности, FIFO (первым вошел, первым вышел) структуру данных.

**Stl\_raw\_storage\_iter.h:**

Файл содержит реализацию класса std::raw\_storage\_iterator из билиотеки STL. Данный класс итератора предоставляет алгоритмам возможность хранить свои результаты в неинициализированной памяти.

**Stl\_relops.h:**

Файл содержит реализацию класса std::relops из библиотеки STL. Данный класс на основе заданных пользователем оператора == и оператора < для объектов реализует обычную семантику других операторов сравнения.

**Stl\_set.h:**

Файл содержит реализацию класса std::set из библиотеки STL. Данный класс представляет собой ассоциативный контейнер, который содержит упорядоченный набор объектов типа “Ключ”.

**Stl\_stack.h:**

Файл содержит реализацию класса std::stack из библиотеки STL, который является контейнерным адаптером, предоставляющим функциональность стека - в частности, FILO (первый вошел, последний вышел) структуры данных.

**Stl\_tempbuf.h:**

Файл содержит реализацию класса \_Temporary\_buffer. Класс предоставляет интерфейс для создания и удаления временного буфера, а также получения его размера, указателей на начало и конец.

**Stl\_tree.h:**

Файл содержит реализацию класса \_Rb\_tree, который представляет собой красно-черное дерево. Данный класс используется при реализации контейнеров типа std::map, std::set, std::multimap и пр.

**Stl\_unititialized.h:**

Файл содержит реализацию классов std::unititialized\_copy, std::unititialized\_fill, std::unitialized\_move.

std::uninitialized\_copy - предоставляет интерфейс для копирования данных из заданного диапазона в неинициализированную ранее память. Все возможные исключения перехватываются, при их возникновении выделенная память освобождается. В противном случае возвращается указатель на выделенную память, в которой находятся скопированные данные.

std::uninitialized\_fill - предоставляет интерфейс для копирования конкретного значения в неинициализированную ранее память, определяемую диапазоном. Все возможные исключения перехватываются, при их возникновении выделенная память освобождается. В противном случае возвращается указатель на выделенную память, в которой находятся скопированные данные.

std::uninitialized\_move - предоставляет интерфейс для перемещения данных из заданного диапазона в неинициализированную ранее память. Все возможные исключения перехватываются, при их возникновении выделенная память освобождается. В противном случае возвращается указатель на выделенную память, в которой находятся скопированные данные.

**Stl\_vector.h:**

Файл содержит реализацию класса std::vector из библиотеки STL. Std::vector представляет собой последовательный контейнер, инкапсулирующий массивы переменного размера.

**Unique\_ptr.h:**

Файл содержит реализацию класса std::unique\_ptr из библиотеки STL. Std::unique\_ptr - умный указатель, который: получает единоличное владение объектом через его указатель, и разрушает объект через его указатель, когда unique\_ptr выходит из области видимости.

**Unordered\_map.h:**

Файл содержит реализацию класса std::unordered\_map из библиотеки STL. Unordered map является ассоциативным контейнером, который содержит пары ключ-значение с уникальными ключами в котором поиск, вставка и удаление выполняются за константное время.

**Unordered\_set.h:**

Файл содержит реализацию класса std::unordered\_set из библиотеки STL. Unordered set представляет собой ассоциативный контейнер, который содержит множество уникальных объектов. Поиск, вставка и удаление в нем осуществляются за константное время.

**Valarray\_after.h:**

Файл содержит реализацию функций sum, min, max, а также оператора получения элемента по индексу для класса std::valarray из библиотеки STL.

**Valarray\_array.h:**

Файл содержит реализацию функций создания массивов, их копирования и перемещения, для класса std::valarray из библиотеки STL.

**Valarray\_before.h:**

Файл содержит реализацию тригонометрических функций (sin, cos, tan и пр.), функций для сравнения массивов, некоторые булевы функции (xor, and, or, и пр.) а также операций сложения, вычитания, деления, взятия остатка, умножения, взятия корня, возведения в степень для класса std::valarray из библиотеки STL. Математические функции применяются к каждому элементу массива valarray.

**/c:**

Содержит файлы-заглушки без исполняемого кода.

**/c\_compatibility:**

Содержит файлы-заглушки без исполняемого кода.

**/c\_global:**

Cписок файлов-заглушек: cassert, ccomplex, cctype, cerrno, cfenv, cfloat, cinttypes, ciso646, climits, cmath, clocale, csetjmp, csignal, cstdalign, cstdarg, cstdbool, cstdint, cstdio, cstdlib, cstring, ctgmath, ctime, cuchar, cwchar, cwctype;

Cstddef - cодержит функции, реализующие побитовые операции.

**/c\_std:**

Содержит файлы-заглушки без исполняемого кода

**/debug:**

**/decimal:**

Файлы decimal и decimal.h содержат функции для конвертации десятичных дробей в типы float, double, long long, математические операции (+, -, и т.д.) и переопределенные операторы сравнения.

**/experimental:**

**/ext:**

**/parallel: (не закончено)**

Cписок файлов-заглушек: algorithm, basic\_iterator.h, compiletime\_settings.h, features.h, algorithmfwd.h(содержит только описания функций), numericfwd.h, parallel.h, types.h, settings.h (содержит различные настройки), tags.h (пустые функции и функции-обертки),

**Algo.h:**

Файл содержит множество функций-оберток для реализации различных алгоритмов. Данные функции в своем теле выполняют только вызов аналогичных функций, уже определенных в других файлах.

**Algobase.h:**

Файл содержит функции-обертки над уже определенными функциями, выполнящими реализацию алгоритмов сравнения

**Balanced\_quicksort.h:**

Файл содержит функции, используемые для реализации алгоритма быстрой сортировки

**Base.h:**

Файл содержит различные функции, реализующие параллельное выполнение базовых математических функций, функций сравнения, шифрование и дешифрование чисел.

**Checkers.h:**

Файл содержит реализацию функции, проверяющую контейнер, для итератора которого определена операция инкремента, на то, является он отсортированным или нет

**Equally\_split.h:**

Файл содержит функции для равного (или примерно равного, если невозможно разбить на равные части) разбиения последовательности

**Find.h:**

Файл содержит функции, реализующие параллельный аналог функции std::find

**For\_each.h:**

Файл содержит функцию-обертку, вызывающую различные вариации функции for\_each\_... в зависимости от типа (тега) метода параллелелизма

**/precompiled:**

Все файлы являются файлами-заглушками.

**/profile:**

**/tr1:**

Список файлов-заглушек: ccomplex, ctype, cfenv, cfloat, cinttypes, climits, cmath, complex.h, cstdarg, cstdbool, cstdint, cstdio, cstdlib, ctgmath, ctime, ctype.h, cwchar, cwctype, fenv, float.h, inttypes.h, limits.h, math.h, memory, random, special\_function\_util.h, stdarg.h, stdbool.h, stdint.h, stdio.h, stdlib.h, tgmath.h, unordered\_map, unordered\_set, wchar.h, wctype.h

**Array:**

Содержит реализацию функций класса std::array – стандартного контейнера для хранения фиксированного числа элементов.

**Bessel\_function.h**

Файл содержит реализацию различных математических функций

**Beta\_function.tcc:**

Файл содержит реализацию различных математических функций

**Complex**

Содержит реализацию различных математических функций

**Ell\_integral.tcc:**

Файл содержит реализацию различных математических функций

**Exp\_integral.tcc:**

Файл содержит реализацию различных математических функций

**Functional:**

Файл содержит большое число функций, часть из которых являются пустыми, а другая часть только возвращает определенное значение. Также содержатся различные шаблоны.

**Functional\_hash.h:**

Файл содержит реализацию функций хеширования для чисел различного типа и строк

**Gamma.tcc:**

Файл содержит реализацию различных математических функций

**Hashtable.h:**

Файл содержит реализацию класса hashtable, предоставляющего набор функций для работы с хеш таблицами (функции вставки , поиска, удаления элементов в контейнере, хеширования, конструкторов, деструктора, оператора присваивания), а также реализацию итератора для этого контейнера

**Hashtable\_policy.h:**

Файл содержит реализацию различных типов итераторов для работы с хеш таблицами, а также вспомогательных функций для работы с контейнерами типa std::map.

**Hypergeometric.tcc:**

Содержит реализацию различных математических функций

**Legendre\_function.tcc:**

Содержит реализацию различных математических функций

**Modified\_bessel\_func.tcc:**

Содержит реализацию различных математических функций

**Poly\_hermite.tcc:**

Содержит реализацию различных математических функций

**Poly\_languerre.tcc:**

Содержит реализацию различных математических функций

**Random.tcc:**

Содержит реализацию различных математических функций

**Regex:**

Файл содержит различные функции для создания регулярных выражений, и операций над ними.

**Riemann\_zeta.tcc:**

Файл содержит реализацию различных математических функций

**Shared\_ptr.h:**

Файл содержит реализацию функций-обертки для работы с классом std:;shared\_ptr.

**Tuple:**

Файл содержит реализацию функций класса std::tuple. Tuple предоставляет интерфейс кортежа. Содержит функции создания кортежа, расформирования его на отдельные объекты, получения информации об элементах, операторы сравнения.

**Type\_traits:**

Файл содержит по большей части пустые функции, а также вспомогательные функции для работы со ссылками, указателями и константами

**Random.h:**

Файл содержит в себе реализацию генератора случайных чисел. Основными классами являются : normal\_distribution, binomial\_distribution, poisson\_distribution, bernoulli\_distribution, cauchy\_distribution, discrete\_distribution, которые реализуют соответственно нормальное, биномиальное распределение, распределение Пуассона, Бернулли, Коши, а также дискретное распределение.

**Unordered\_map.h:**

Файл содержит пустые функции-заглушки и реализацию функции swap(a,b) для контейнеров unordered\_map и unordered\_multimap, которая в теле выполняет вызов уже определенного для этих классов метода swap (a.swap(b))

**Unordered\_set.h:**

Аналогично с файлом Unordered\_map.h для контейнеров unordered\_set и unordered\_multiset

**Utility:**

Файл содержит вспомогательные функции для преобразования std::pair в std::tuple

**/tr2:**

**Bool\_set:** содержит реализацию класса bool\_set, в который входят следующие функции: конструкторы, деструктор, функции сравнения, логические функции, функции приведения в булевский тип, проверки на пустоту, функции для работы с множествами.

**Dynamic\_bitset:** содержит реализацию соответствующего класса, представляющего собой версию контейнера std::bitset, способную изменять свой размер во время выполнения программы.

Файлы-заглушки : Ratio, type\_traits

**Libsupc++**

Список файлов-заглушек: array\_type\_info.cc, atexit\_arm.cc, atomic\_lockfree\_defines.h, bad\_alloc.cc, bad\_array\_length.cc, bad\_array\_new.cc, bad\_cast.cc, bad\_typeid.cc, cxxabi.h, cxxabi\_forced.h, del\_op.cc, del\_opnt.cc, del\_ops.cc, del\_opv.cc, del\_opvnt.cc, del\_opvs.cc, eh\_aux\_runtime, eh\_exception.cc, eh\_term\_handler.cc, eh\_type.cc, eh\_unex\_handler.cc, enum\_type\_info.cc, exceptions\_defines.cc, function\_type\_info.cc, guard\_error.cc, hash\_bytes.h, nested\_exception.cc, new\_handler.cc, new\_op.cc, new\_opnt.cc, new\_opv.cc, new\_opvnt.cc, pbase\_type\_info.cc, pmem\_type\_info.cc, pointer\_type\_info.cc, pure.cc, tinfo2.cc, vtv\_stubs.cc

**Po**

Содержит файл без исполняемого кода string\_literals.cc

**Src**

**/C++11**

Файлы-заглушки либо файлы без исполняемого кода: сompatibility-chrono.cc, compatibility-condvar.cc, compatibility-thread-c++0x.cc, cow-stdexcept.cc, cow-locale\_init.cc, cow-fstream\_inst.cc, cow-shim\_facets.cc, cow-sstream-inst.cc, сow-string-io-inst.cc, cow-wstring-inst.cc, cctype.cc, cow-string-inst.cc, cow-wstring-io-inst.cc, cxx11-hash\_tr1.cc, cxx11-ios\_failure.cc, cxx11-locale-inst.cc, cxx11-stdexcept.cc, cxx11-wlocale-inst.cc, ext11-inst.cc, fstream-inst.cc, functexcept.cc, functional.cc, ios-inst.cc, iostream-inst.cc, istream-inst.cc, limits.cc, locale-inst.cc, ostream-inst.cc, mutex.cc, mutex\_pool.h, placeholders.cc, regex.cc, sso-string.cc, system-error.cc, string-io-inst.cc, string-inst.cc, sstream-inst.cc, streambuf-inst.cc, wlocale-inst.cc, wstring-inst.cc, wstring-io-inst.cc

**Chrono.cc:**

Файл содержит реализацию функции now из класса std::chrono, которая показывает текущее время

**Codecvt.cc:**

Файл содержит реализацию класса codecvt, и содержит функции для перевода строк в различные кодировки, в том числе и многобайтовые.

**Compatibility-atomic-c++0x.cc:**

Файл содержит реализацию класса std::atomic\_flag, который представляет атомарные функции для установки флага в состояния true и false.

**Compatibility-c++0x.cc:**

Файл содержит реализацию классов lock\_error (заглушка), error\_category, system\_clock (заглушка), hash (функтор).

Error\_category – базовый виртуальный класс для определенных категорий ошибок, например std::system\_category, std::iostream\_category и т.д.

**Futex.cc:**

Содержит реализацию двух функций: m\_futex\_wait\_until (функция-таймер, ожидает в течение определенного времени) , m\_futex\_notify\_all (обертка над системным вызовом)

**Future.cc:**

Файл содержит реализацию класса future\_error\_category, наследованного от вышеописанного класса error\_category

**Hash\_c++0x.cc:**

Файл содержит реализацию функции хеширования чисел типа long double.

**Hashtable\_c++0x.cc:**

Файл содержит две вспомогательные функции для хеширования, служащие для поиска простых чисел.

**Condition\_variable.cc:**

Файл содержит реализацию функций для класса condition\_variable: конструктор, деструктор, функции wait, notify\_one, notify\_all. Condition\_variable является примитивом синхронизации, применяющимся для блокировки одного или нескольких потоков, пока от переменной не поступит сигнал о разблокировке.

Wait – блокирует текущий поток, пока переменная не даст сигнал;

Notify\_one - испускает сигнал для одного ожидающего потока;

Notify\_all – испускает сигнал для всех ожидающих потоков;

**Random.cc:**

Файл содержит реализацию следующих функций для класса std::random\_device: функции инициализации, получения значения, получения энтропии.

**Shared\_ptr.cc:**

Файл содержит реализацию конструкторов и деструктора для класса Sp\_locker, а также функцию получения мьютекса.

**Thread.cc:**

Файл содержит реализацию функций Join, detach, start\_thread, hardware\_concurrency для класса std::thread.

Join – функция, которая ожидает завершения потока

Detach – функция, которая позволяет выполнять поток независимо от дескриптора

Start\_thread – функция, создающая новый поток.

Hardware\_concurrency – возвращает количество одновременно выполняющихся потоков

**Snprintf\_lite.cc:**

Содержит функции для перехвата ошибок при форматированном выводе текста свзанных с нехваткой места в буфере, а также вспомогательных функций для вывода и подстановки значений.

**/C++98**

Список файлов-заглушек и файлов не содержащих исполняемого кода:

allocator-inst.cc, compatibility-debug\_list.cc, compatibility-debug\_list-2.cc, compatibility-ldbl.cc, complex\_io.cc, concept-inst.cc, codecvt.cc, cow-istream-string.cc, ext-inst.cc, globals\_io.cc, hashtable\_tr1.cc, hash\_tr1.cc, ios\_failure.cc, ios\_locale, list-associated.cc, list-associated-2.cc, math\_stubs\_float.cc, math\_stubs\_long\_double.cc, locale\_init.cc, misc-inst.cc, parallel\_settings.cc, valarray.cc, stdexcept.cc

**Bitmap\_allocator.cc:**

Файл содержит реализацию функций, отвечающих за выделение и освобождение памяти для класса free\_list.

**Compatibility.cc:**

Файл содержит реализацию функции ignore из класса basic\_istream, которая выполняет чтение из потока либо некоторого количества символов, либо до разделителя.

**Ios\_init.cc:**

Файл содержит реализацию конструктора, деструктора и функции sync\_with\_stdio из класса ios\_base.

Sync\_with\_stdio выполняет проверку синхронизации стандартных потоков ввода и вывода С++ со стандартными потоками С после каждой операции ввода и вывода

**Istream-string.cc:**

Файл содержит реализацию оператора ввода (>>) и функции getline, считывающую строку из входного потока

**Istream.cc:**

Файл содержит реализацию функций Getline, ignore, оператора ввода >> с различными параметрами.

**List-aux-2.cc:**

Файл содержит реализацию функций transfer, reverse, hook, unhook класса List\_node\_base

Transfer: перемещает несколько элементов в указанном диапазоне в другую позицию списка

Reverse: функция выполняет реверс списка

Hook: функция выполняет вставку элемента в список

Unhook: функция убирает элемент из списка

**List-aux.cc:**

Файл содержит реализацию функций Transfer, reverse, hook, unhook, swap из класса list\_node\_base. Реализация первых четырех функций совпадает с реализацией из файла List-aux-2.cc

Swap: функция меняет местами заданные элементы списка

**Hash-long-double-tr1-aux.cc:**

Файл содержит реализацию функции хеширования чисел типа long double

**Locale.cc:**

Файл содержит реализацию класса locale: конструкторы, деструктор, оператор сравнения (==), оператор присваивания, функцию получения имени локали (name), а также функции для инициализации, добавления и получения новых фасетов.

**Locale\_facets.cc:**

Файл содержит функции для группировки переданных данных в фасеты, а также различные вспомогательные константы

**Localename.cc:**

Файл содержит реализацию функций для инициализации локалей, а также управления категориями локалей и их замены.

**Strstream.cc:**

Файл содержит реализацию функций классов strstream, ostrstream, istrstream, strstreambuf. Реализованы следующие функции:

Конструкторы, деструкторы

Freeze : функция включает либо отключает автоматическое выделение и освобождение памяти для буфера

Str : функция для доступа к выходному буферу

Pcount: функция возвращает количество введенных символов

Rdbuf: функция возвращает указатель типа strstreambuf на связанный буфер

Seekoff: функция для изменения позиции указателя во входной/выходной последовательности

Undeflow: функция для чтения символа из входной последовательности

Pbackfail: возвращает прочитанный символ обратно во входную последовательность

Overflow: функция для довыделения памяти, в случае когда это необходимо и возможно

**Strambuf.cc:**

Файл содержит две реализации одной функции copy\_streambufs\_eof для разных типов аргументов. Функция выполняет копирование буфера.

**Pool\_allocator.cc:**

Файл содержит реализацию функций для аллокации памяти входящих в класс pool\_alloc\_base

**Mt\_allocator.cc:**

Файл содержит реализацию функций для управления памятью, её выделения, резервирования и освобождения, реализацию класса freelist, который управляет списом блоков памяти.

**Tree.cc:**

Файл содержит базовые функции, используемые в реализации красно-черного дерева, такие как функции вставки, перебалансровки, удаления, поворота дерева, а также функции rb\_tree\_increment и rb\_tree\_decrement, реализующие соответственно операции инкремента и декремента по элементам дерева.

**/Filesystem**

Список файлов-заглушек: cow-dir.cc, cow-ops.cc, cow-path.cc, cow-std-dir.cc, cow-std-path.cc, ops-common.h, cow-std-ops.cc, std-dir.cc, std-ops.cc, std-path.cc (в трех последних файлах аналогично описаны те же функции, что и в файлах ops.cc, dir.cc, path.cc)

**Dir-common.h:**

Файл содерджит реализацию функций класса Dir\_base : конструкторы, деструктор, функции advance (выполняет переход в директорию на уровень ниже, если это возможно) и is\_dor\_or\_dotdot (возвращает результат сравнения переданной в качестве аргумента строки с символом точки или двух точек), также содержится реализация функции get\_file\_type, возвращающая тип файла, переданного ей в качестве аргумента.

**Dir.cc:**

Файл содержит реализацию оберток над функцией advance, описанной выше. Данные реализации выполняют вызов этой функции из файла Dir-common.h. Также содержатся реализации функций класса directory\_iterator (однонаправленный итератор), позволяющие итерироваться по директориям. Работа этих функций основана на работе функции advance из файла dir-common.h

**Ops.cc:**

Файл содержит реализацию функций из класса fs, служащих для работы с директорями и файлами : получение информации о файле (права, размер, проверка на пустоту, время последнего изменения), создание и удаление файлов, поулчение абсолютного и относительного пути, получение списка файлов в директории, копирование файлов, создание директорий.

**Path.cc:**

Файл содержит реализациюю функций класа path: конструкторы, деструктор, операторы сравнения, функции удаления файлов, получения их имени, получения относительного пути, получения пути к корневой директории файловой системы.

/**Shared**

Директория содержит файл hashtable-aux.cc содержищий extern константу (список простых чисел)

**Libgcc**

Список файлов-заглушек либо файлов, не содержащих исполняемого кода: Vtv\_start\_preinit.c, vtv\_start.c, vtv\_end\_preinit.c, vtv\_end.c, offloadstuff.c, sync.c, unwind-compat.h, gthr.h, gstdint.h, enable-execute-stack-emty.c

**Crtstuff.c:**

Файл содержит различные функции для запуска конструкторов при старте программы и деструкторов при её завершении

**Dfp-bit:**

Файл содержит реализацию различных бинарных операций, конвертеров, а также реализацию математических функций для чисел типа float

**Divmod.c:**

Файл содержит две функции-обертки, вызывающие функции для получения целой части и остатка от деления (div, mod)

**Enable-execute-stack-mprotect.c:**

Содержит функцию проверки возможности использования стека, и функцию, которая включает возможность его использования

**Fixed-bit:**

Файл содержит реализацию различных битовых операций и функций для преобразования типов

**Floatunsisf.c, floatunsisf.c, floatunsitf.c, floatunsixf.c :**

Файлы содержат функции для конвертации чисел типа int в типы float, double

**Fp-bit:**

Файл содержит реализацию математических функций, функций для приведения типов, сравнения

**Gbl-ctors.h:**

Файл содержит описания функций (а также определение одной функции), вызываемых при старте программы, выполняющих вызов конструкторов. Данные функции используются в файлах libgcc2.c и crtstuff.c

**Generic-morestack.c:**

Файл содержит различные функции для работы со стеком (аллокация памяти, освобождение и пр.)

**Generic-morestack.h:**

Файл содержит объявления функций для аллокации, освобождения памяти, перехвата ошибок

**Generic-morestack-thread.c:**

Файл содержит объявления и описания функций для работы с потоками pthread

**Gthr-posix.h:**

Файл содержит функции-обертки для работы с потоками и мьютексами

**Gthr-single.h:**

Файл содержит функции-обертки и функции-заглушки для работы с потоками и мьютексами

**Libgcc2:**

Файл содержит реализацию математических функций (сложение, вычитание, деление, возведение в степень и т.д.) для различных типов и функций конвертации

**Libgcov.h:**

Файл содержит только объявления (без определений) различных структур и функций

**Libgcov-driver.c:**

Файл содержит реализацию функций, необходимых для работы программы. Это функции для вывода ошибок, инициализации объектов, их уничтожения, функции, вызывающиеся при старте и выходе из программы.

**Libgcov-driver-system.c:**

Файл содержит функции для создания директорий, файлов, и открытия файлов в формате .gcda

**Libgcov-interface.c:**

Файл содержит функции-обертки для работы с gcov файлами (они используются для хранения информации о профилировке)

**Libgcov-merge.c:**

Файл содержит реализацию функций для профилировки, подсчета количества вызовов функций.

**Libgcov-profiler.c:**

Файл содержит реализацию вспомогательных функций для профилировки

**Libgcov-util.c:**

Файл содержит реализацию вспомогательных функций для работы с gcov файлами, их создания и вычисления статистики

**Memcmp.c:**

Файл содержит реализацию функции memcmp

**Memcpy.c:**

Файл содержит реализацию функции memcpy

**Memmove.c:**

Файл содержит реализацию функции memmove

**Memset.c:**

Файл содержит реализацию функции memset

**Udivmod.c:**

Файл содержит две функции-обертки, выполняющие вызов функции udivmodsi4

**Udivmodsi4.c:**

Файл содержит реализацию функции udivmodsi4, выполняющей деление и возвращающей либо остаток от деления либо целую часть

**Unwind-c.c:**

**Unwind-compat:**

Файл содержит функции-обертки которые в своем теле выполняют только вызов другой функции с аналогичным названием и возвращают результат её выполнения.

**Unwind-dw2:**

**Unwind-dw2-fde:**

**Unwind-dw2-fde-compat.c:**

**Unwind-dw2-fde-dip.c:**

**Unwind-generic.h:**

**Unwind-pe.h:**

**Unwind-seh.c:**

**Unwind-sjlj.c:**

**/soft-fp**

Во всех .с файлах содержится реализация функции для работы с числами с плавающей точкой, название которой соответствует названию файла. При этом в теле функции содержатся вызовы различных макросов, которые определены в .h файлах.