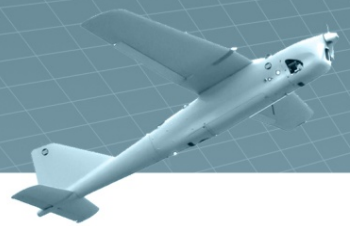
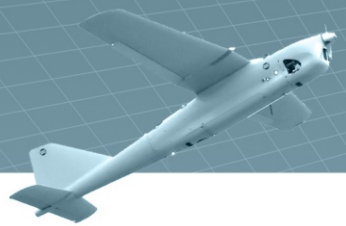
Фильченко Н.В.

C++: STL часть 1



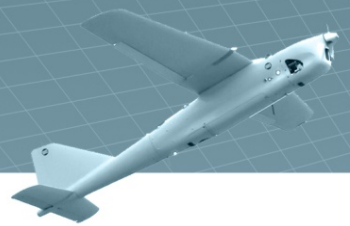
STL

- Утилиты
  - Управление памятью
  - Numeric limits
  - Обработка ошибок
- Работа со строками
- Контейнеры
- Алгоритмы
- Итераторы
- Ввод/вывод
- Локализация
- Регулярнве выражения
- Атомарные операции
- Потоки
- Совместимость с C



Утилиты

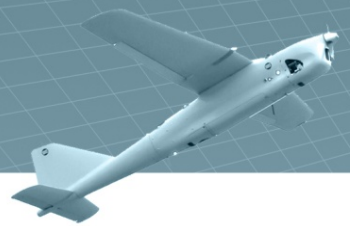
- **<cstdlib>** -- Замена stdlib.h
- **<csignal>** -- Замена signal.h
- **<csetjmp>** -- Замена setjmp.h
- **<cstdarg>** -- Замена stdarg.h
- **<typeinfo>** -- Информация о типах во время выполнения
- **<typeindex>** -- std::type_index
- **<type_traits>** -- Информация о типах во время компиляции
- **<bitset>** -- std::bitset
- **<functional>** -- Функторы
- **<utility>** -- Утилиты
- **<ctime>** -- Замена time.h
- **<chrono>** -- Работа с временем
- **<cstddef>** -- Замена stddef.h
- **<initializer_list>** -- Списки инициализации
- **<tuple>** -- std::tuple



typeinfo

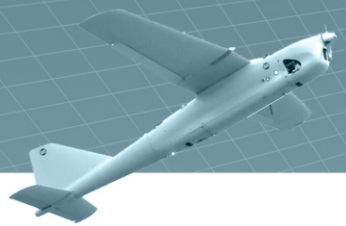
Типы

- **typeinfo** -- тип результата оператора typeid
- **bad_typeid** -- исключение бросается, если аргумент typeid является NULL
- **bad_cast** -- исключение бросается, если аргумент dynamic_cast ссылка и приведение невозможно



std::typeinfo

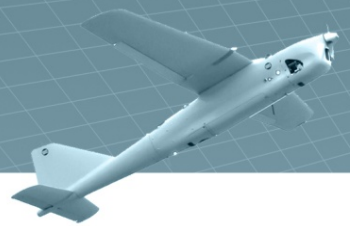
```
class type_info {
public:
    virtual ~type_info();
    bool operator==(const type_info& rhs) const noexcept;
    bool operator!=(const type_info& rhs) const noexcept;
    bool before(const type_info& rhs) const noexcept;
    size_t hash_code() const noexcept;
    const char* name() const noexcept;
    type_info(const type_info& rhs) = delete;
    type_info& operator=(const type_info& rhs) = delete;
};
```



typeindex

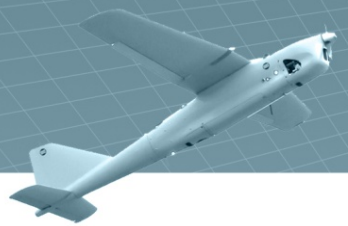
Типы

- **typeindex** -- обертка для std::type_info, которую можно использовать в ассоциативных контейнерах
- **std::hash<std::typeindex>** -- специализация std::hash
- **bad_cast** -- исключение бросается, если аргумент dynamic_cast ссылка и приведение невозможно



std::typeindex

```
class type_index {
public:
    type_index(const type_info& rhs) noexcept;
    bool operator==(const type_index& rhs) const noexcept;
    bool operator!=(const type_index& rhs) const noexcept;
    bool operator< (const type_index& rhs) const noexcept;
    bool operator<= (const type_index& rhs) const noexcept;
    bool operator> (const type_index& rhs) const noexcept;
    bool operator>= (const type_index& rhs) const noexcept;
    size_t hash_code() const;
    const char* name() const;
};
```



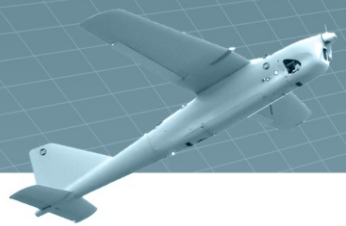
type_traits

Вспомогательные типы

- **integral_constant** -- константа времени компиляции заданного типа
- **true_type** -- std::integral_constant<bool, true>
- **false_type** -- std::integral_constant<bool, false>

Основные категории

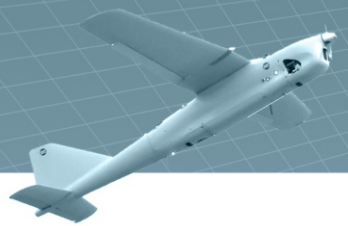
- **is_void** -- Проверка на void
- **is_null**$_p$*ointer* — —*std* :: *nullptr*$_t$**is_integral** — —
- **is_floating_point** -- Проверка на тип с плавающей точкой
- **is_array** -- Проверка на массив
- **is_enum** -- Проверка на перечисляемый тип
- **is_union** -- Проверка на объединение
- **is_class** -- Проверка на класс/структуру
- **is_function** -- Проверка на функцию
- **is_pointer** -- Проверка на указатель
- **is_lvalue_reference** -- Проверка на lvalue
- **is_rvalue_reference** -- Проверка на rvalue
- **is_member_object_pointer** -- Проверка на указатель на поле
- **is_member_function_pointer** -- Проверка на указатель на метод



integral_constant

```
template< class T, T v >
struct integral_constant;

template <bool B>
using bool_constant = integral_constant<bool, B>
```



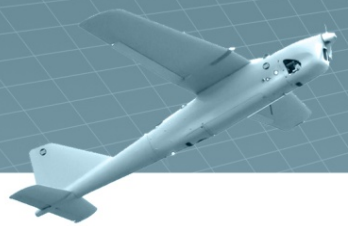
type_traits

Составные категории

- **is_fundamental** -- Проверка на фундаментальный тип
- **is_arithmetic** -- Проверка на численный тип
- **is_scalar** -- Проверка на скаляр
- **is_object** -- Проверка на объект
- **is_compound** -- Проверка на составной тип
- **is_reference** -- Проверка на ссылку
- **is_member** -- Проверка на указатель на член

Свойства типов

- **is_const** -- Проверка на константность
- **is_volatile** -- Проверка на volatile
- **is_trivial** -- Проверка на тривиальность
- **is_trivially_copyable** -- Проверка на тривиальное копирование
- **is_standard_layout** -- Проверка на STDLayout
- **is_pod** -- Проверка на POD
- **is_literal** -- Проверка на literal
- **is_empty** -- Проверка на отсутсвие полей
- **is_polymorphic** -- Проверка на полиморфизм
- **is_abstract** -- Проверка на абстрактность
- **is_signed** -- Проверка на знак
- **is_unsigned** -- Проверка на знак



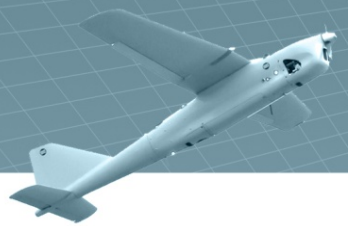
type_traits

Существование конструктора

- is_constructible
- is_trivially_constructible
- is_nothrow_constructible
- is_default_constructible
- is_trivially_default_constructible
- is_nothrow_default_constructible
- is_copy_constructible
- is_trivially_copy_constructible
- is_nothrow_copy_constructible
- is_move_constructible
- is_trivially_move_constructible
- is_nothrow_move_constructible

Проверка присвоения

- is_assignable
- is_trivially_assignable
- is_nothrow_assignable
- is_copy_assignable
- is_trivially_copy_assignable
- is_nothrow_copy_assignable
- is_move_assignable
- is_trivially_move_assignable
- is_nothrow_move_assignable



type_traits

Проверка деструктора

- **is_destructible**
- **is_trivially_destructible**
- **is_nothrow_destructible**
- **has_virtual_destructor**

Свойства типа

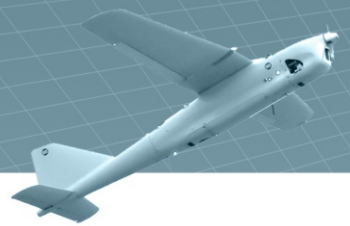
- **alignment_of** -- Требуемое выравнивание
- **rank** -- Число размерностей массива
- **extent** -- Размер массива по заданному измерению

Отношения типов

- **is_same** -- Проверка на равенство
- **is_base** -- Проверка наследования
- **is_convertible** -- Проверка на возможность преобразования

Изменение const/volatile

- **remove_cv**
- **remove_const**
- **remove_volatile**
- **add_cv**
- **add_const**
- **add_volatile**



type_traits

Изменение ссылок

- **remove_reference** -- Преобразование ссылки в тип
- **add_lvalue_reference** -- Преобразование типа к lvalue ссылке
- **add_rvalue_reference** -- Преобразование типа к rvalue ссылке

Изменение указателей

- **remove_pointer** -- Преобразование указателя к типу
- **add_pointer** -- Преобразование типа к указателю

Изменение знака

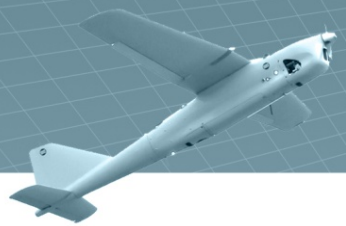
- **make_signed** -- Преобразование к знаковому типу
- **make_unsigned** -- Преобразование к беззнаковому типу

Изменение массива

- **remove_extent** -- Удаление одной размерности
- **remove_all_extents** -- Удаление всех размерностей

Прочие изменения

- **aligned_storage** -- Заглушка заданного типа
- **aligned_union** -- Заглушка заданного объединения
- **decay** -- Преобразование типа как при передаче в функцию по значению
- **enable_if** -- заданный тип, если bool аргумент true
- **conditional** -- выбор типа в зависимости от bool
- **common_type** -- тип результата арифметического выражения
- **underlying_type** -- целочисленный тип перечисления
- **result_of** -- тип возвращаемого значения



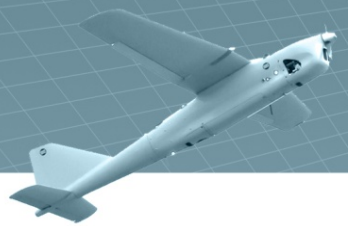
bitset

Типы

- **std::bitset** -- Набор битовых флагов
- **std::hash<bitset>** -- Специализация std::hash

Функции

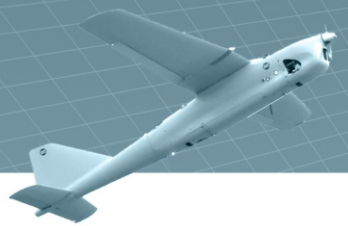
- **operator&**
- **operator|**
- **operator^**
- **operator<<**
- **operator>>**



std::bitset

```
template<std::size_t N>
class bitset;
```

- Позволяет работать с наборами бит как с массивом bool
- Размер фиксирован



functional

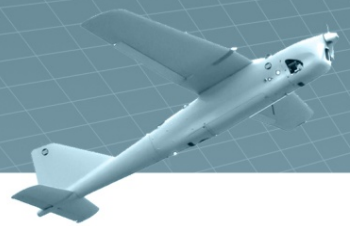
Типы

- **std::function** -- Заворачивает любой вызываемый объект
- **std::mem_fn** -- Заворачивает указатель на метод класса
- **std::bad_function_call** -- Исключение бросается при вызове пустого std::function
- **std::is_bind_expression** -- Проверка на bind выражение
- **std::is_bind_placeholeder** -- Проверка на заглушку (_1, _2…)
- **std::reference_wrapper** -- Обертка для ссылок
- **std::placeholders::_1**,**std::placeholders::_2**…

Функции

- **std::bind** -- Назначение одного или нескольких аргументов функции
- **std::ref** -- создание std::reference_wrapper
- **std::cref** -- создание std::reference_wrapper

Также есть function ojects (and, or, …)



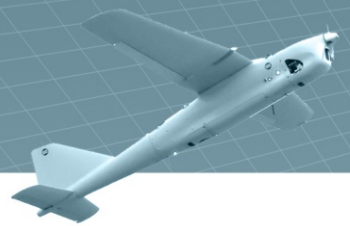
utility

Типы

- **std::pair** -- Пара
- **std::integer_sequence** -- Последовательность целых веремени компиляции

Функции

- **std::swap** -- Обмен двух ссылок значениями
- **std::exchange** -- Обновление занчения и получение старого
- **std::forward** -- Передача аргумента следующей функции
- **std::move** -- Получение rvalue ссылки
- **std::move_if_noexcept** -- Получение rvalue ссылки, если это не вызовет исключений. Иначе lvalue
- **std::declval** -- Получения lvalue типа, не имеющего конструктора по умолчанию
- **std::make_pair** -- Создание пары
- **std::make_get** -- Получение элемента пары



tuple

Типы

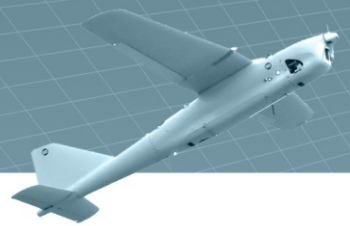
- **std::tuple**
- **std::tuple_size** -- Получение количества элементов std::tuple
- **std::tuple_element** -- Получение типа элемента std::tuple

Константы

- **std::ignore** -- Пропус элемента для std::tie

Функции

- **std::make_tuple** -- Создание std::tuple
- **std::tie** -- Распаковка std::tuple
- **std::forward_as_tuple** -- Передача rvalue параметров как tuple
- **std::tuple_cat** -- Объединение std::tuple
- **std::get** -- Получение элемента std::tuple



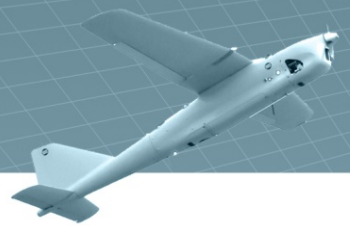
initializer_list

Типы

- **std::initializer_list<T>** -- Список инициализации с элементами типа T

Функции

- **std::begin<std::initializer_list>** -- специализация std::begin
- **std::begin<std::initializer_list>** -- специализация std::end



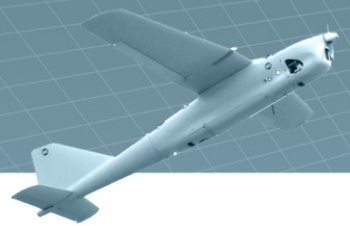
chrono

Типы

- **std::chrono::duration** -- Промежуток времени
- **std::chrono::system_clock** -- Системные часы
- **std::chrono::steady_clock** -- Монотонные часы
- **std::chrono::high_resolution_clock** -- Максимально точные часы
- **std::chrono::time_point** -- Время
- **std::chrono::duration_values** -- zero, min, max

Функции

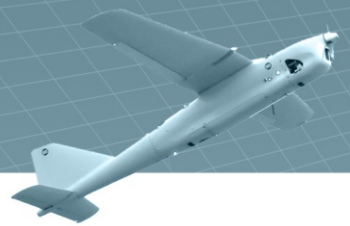
- **std::chrono::time_point_cast** -- Преобразование времени
- **std::chrono::duration_cast** -- Преобразование промежутка времени



Задание

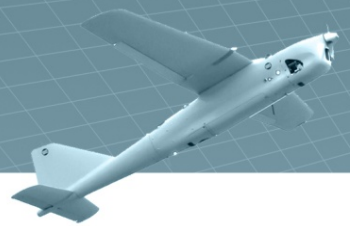
Усовершенствовать format, добавив формат вывода %@, который:

- Если аргумент - nullptr_t -- выводит nullptr
- Если аргумент указатель, и его значение равно 0 -- выводит nulltpr<имя_типа>
- Если аргумент указатель, и его значение не равно 0 - выводит ptr<имя_типа>(вывод значения как для %@)
- Если аргумент массив известной размерности -- выводит элементы массива через запятую в []
- Если аргумент может быть преобразован к std::string -- выводит результат такого преобразования
- Если ни одно преобразование невозможно -- кидается исключение



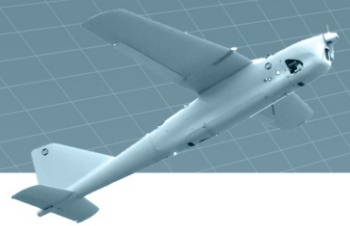
Совместимость с C

- **<ciso646>** -- пустой файл
- **<ccomplex>** -- Подключает **<complex>**
- **<ctgmath>** -- Подключает **<ccomplex>** и **<cmath>**
- **<cstdalign>** -- содержит константу __alignas_is_defined
- **<cstdbool>** -- содержит константу __bool_true_false_are_defined



Управление памятью

- **<new>** -- Низкоуровневое управление памятью
- **<memory>** -- Высокоуровневое управление памятью
- **<scoped_allocator>** -- Многоуровневый менеджер памяти



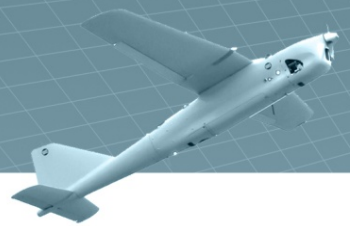
new

Функции

- **operator new**
- **operator new[]**
- **operator delete**
- **operator delete[]**
- **std::new_handler get_new_handler()** -- получить текущий new handler либо nullptr
- **std::new_handler set_new_handler(std::new_handler new_p)** -- установть new handler

Классы

- **std::bad_alloc**
- **std::bad_array_new_length**



memory

Умные указатели

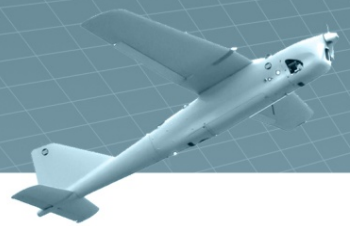
- **std::unique_ptr**
- **std::shared_ptr**
- **std::weak_ptr**

Типы для работы с умными указателями

- **std::enable_shared_from_this**
- **std::bad_weak_ptr**

Функции для работы с умными указателями

- **std::make_unique**
- **std::make_shared**
- **std::allocate_shared**
- **std::static_pointer_cast**
- **std::dynamic_pointer_cast**
- **std::const_pointer_cast**
- **std::reinterpret_pointer_cast**



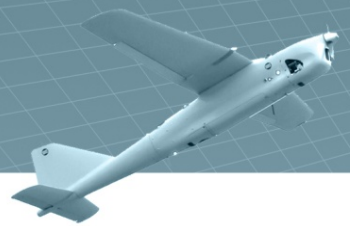
memory

Аллокаторы

- **std::allocator**

Функции

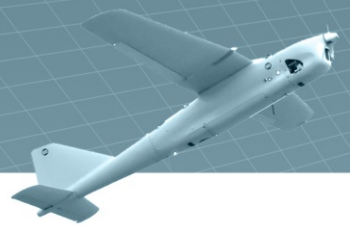
- **std::uninitialized_copy**
- **std::uninitialized_copy_n**
- **std::uninitialized_fill**
- **std::uninitialized_fill_n**
- **std::get_temporary_buffer**
- **std::return_temporary_buffer**



memory

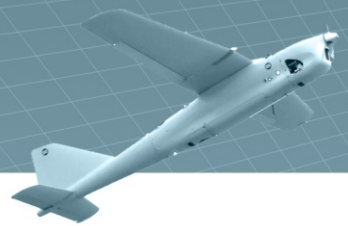
Сборщик мусора

- **void declare_reachable(void* p)**
- **template<class T> T* undeclare_reachable(T* p)**
- **void declare_no_pointers(char *p, std::size_t n)**
- **void undeclare_no_pointers(char *p, std::size_t n)**
- **std::get_pointer_safety** -- strict, preferred, relaxed



Numeric limits

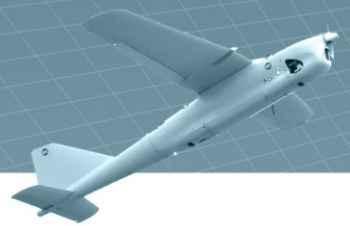
- **<climits>** -- замена limits.h
- **<cfloat>** -- замена float.h
- **<cstdint>** -- замена stdint.h
- **<cinttypes>** -- замена inttypes.h
- **<limits>** -- Свойства арифметический типов



std::numeric_limits<T>

Константы

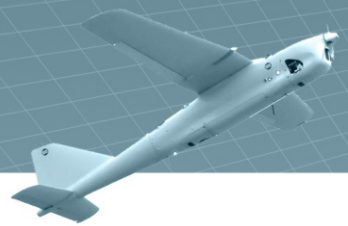
- **is_specialized** -- доступна ли специальизация
- **is_signed** -- проверка на знак
- **is_integer** -- проверка на целочисленный тип
- **is_exact** -- проверка на точный тип
- **has_infinity** -- есть бесконечность
- **has_quiet_NaN** -- есть NaN
- **has_signaling_NaN** -- есть signalling NaN
- **has_denorm** -- есть денормализованное представление
- **is_modulo** -- арифметика по модулю
- **digits** -- количество разрядов в radix системе
- **digits10** -- количество разрядов в десятичной системе



std::numeric_limits<T>

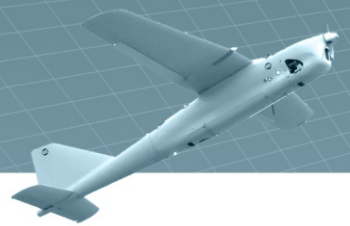
Функции

- **min** -- минимальное нормализованное значение
- **max** -- максимальное значение
- **lowest** -- максимальное значение
- **epsilon**
- **infinity**
- **round_error** -- максимальная ошибка округления



Обработка ошибок

- **<exception>** -- обработка исключений
- **<stdexcept>** -- стандартные типы исключений
- **<cassert>** -- замена assert.h
- **<system_error>** -- std::error_code
- **<cerrno>** -- замена errno.h



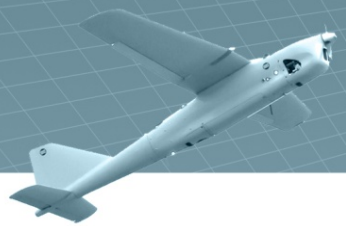
exception

Типы

- **std::exception** -- базовый тип для исключений
- **std::nested_exception** -- тип для захвата вложенного исключения

Функции

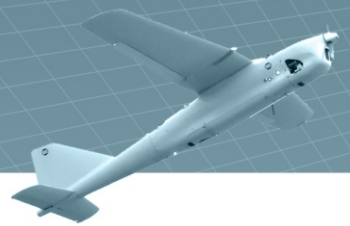
- **std::throw_with_nested**
- **std::current_exception** -- получение указателя на текущее исключение
- **bool uncaught_exception()** -- проверка наличия исключений
- **int uncaught_exceptions()** -- проверка наличия исключений



stdexcept

Стандартные исключения

- **std::logic_error** -- нарушение инварианта
- **std::invalid_argument** -- неверный аргумент
- **std::domain_error** -- вызов функции вне области определения
- **std::length_error** -- превышение допустимой длины
- **std::out_of_range** -- выход за границы
- **std::runtime_error** -- ошибка времени выполнения
- **std::range_error** -- ошибка диапазона
- **std::overflow_error** -- численное переполнение
- **std::underflow_error** -- численное переполнение (слишком маленькое значение)



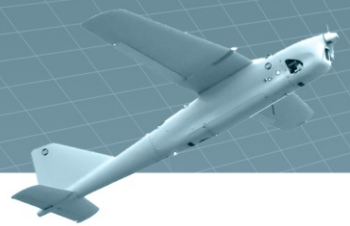
system_error

Типы

- **std::error_category**
- **std::generic_category**
- **std::system_category**
- **std::error_condition** -- переносимый код ошибки
- **std::errc** -- enum из констант <cerrno>
- **std::error_code** -- платформо-зависимый код ошибки
- **std::system_error** -- исключение

Функции

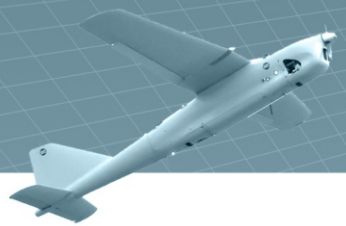
- **std::make_error_code**
- **std::make_error_condition**



Работа со строками

Файлы

- **<cctype>** -- замена ctype.h
- **<cwctype>** -- замена wctype.h
- **<cstring>** -- замена string.h
- **<cwchar>** -- замена wchar.h
- **<cuchar>** -- работа с юникодом
- **<string>** -- строки



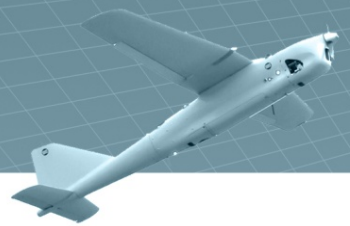
string

Типы

- **std::string**
- **std::wstring** -- строка из wchar_t
- **std::u16string** -- строка из char16_t
- **std::u32string** -- строка из char32_t
- **std::hash<>** -- специализации для строк

Функции

- **std::getline** -- чтение строки из istream
- **std::to_string** -- преобразование числа в строку
- **stoi, stol, stoll, stoul, stoull, stof, stod, stold** -- преобразование строки в число



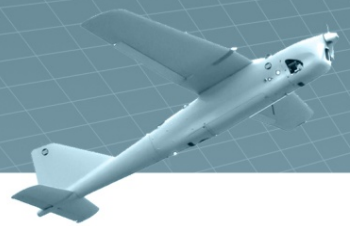
std::string

Доступ

- at
- operator[]
- front
- back
- data
- c_str

Итераторы

- begin, end
- cbegin, cend
- rbegin, rend
- crbegin, crend



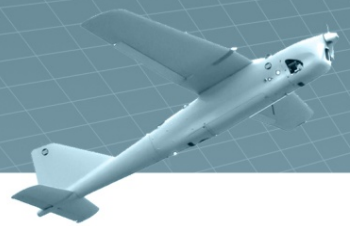
std::string

Размер

- **empty** -- проверка на пустоту
- **size, length** -- длина строки
- **max_size** -- максимально возможная длина строки
- **reserve** -- зарезервировать место
- **capacity** -- количество зарезервированного места
- **shrink_to_fit** -- освободить лишнюю память

Поиск

- **find, rfind** -- поиск символа или строки
- **find_first_of, find_last_of** -- поиск символа из набора
- **find_first_not_of, find_last_not_of** -- поиск символа не из набора



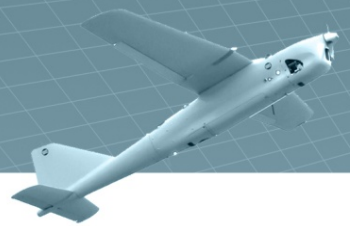
std::string

Операции

- **clear** -- очистка
- **insert** -- вставка
- **erase** -- удаление фрагмента
- **push_back** -- запись в конец
- **append** -- запись в конец
- **pop_back** -- удаление последнего символа
- **compare** -- сравнение строк
- **replace** -- замена фрагмента
- **substr** -- подстрока
- **copy** -- подстрока
- **resize** -- изменить размер

Константы

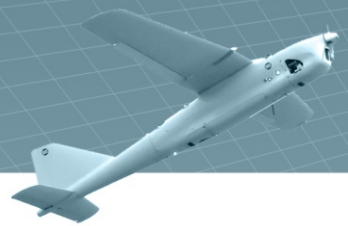
- **npos** -- специальное значение позиции (-1)



Задание

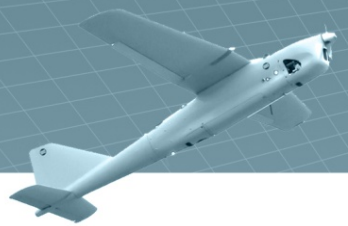
Класс lazy_string, реализующий copy-on-write

- Конструируется от std::string, приводится к std::string
- Имеет size, length, at, operator[] аналогично std::string
- Имеет операторы ввода и вывода в поток (<< >>)
- Имеет substr, причем при взятии подстроки копирования не происходит

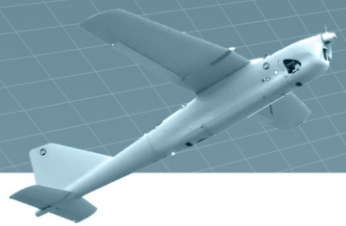


Контейнеры

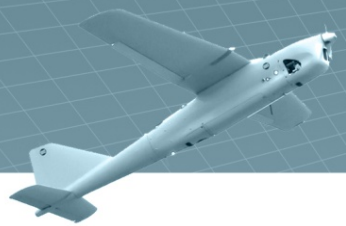
- **<array>** -- массив фиксированного размера
- **<vector>** -- мектор
- **<dequeue>** -- дек
- **<list>** -- двусвязный список
- **<forward_list>** -- односвязный список
- **<set>** -- множество на основе дерева
- **<map>** -- словарь на основе дерева
- **<unordered_map>** -- словарь на основе хэш=таблицы
- **<unordered_set>** -- множество на основе хэш-таблицы
- **<stack>** -- стек
- **<queue>** -- очередь



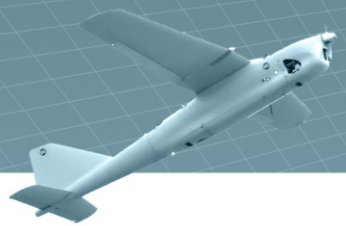
<array>



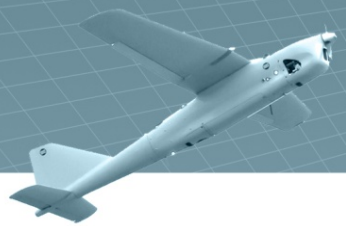
<vector>



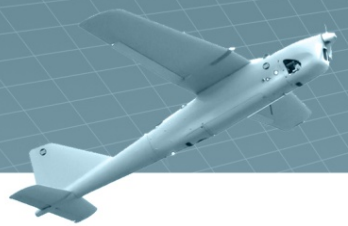
<dequeue>



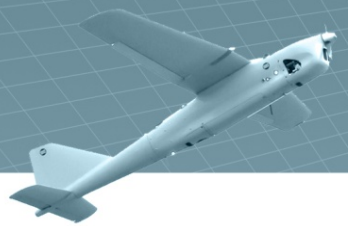
<list>



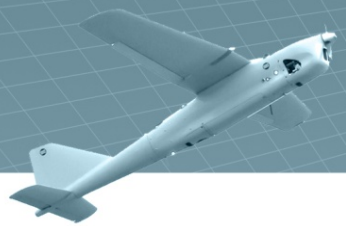
<forward_list>



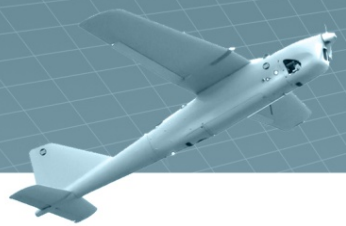
<set>



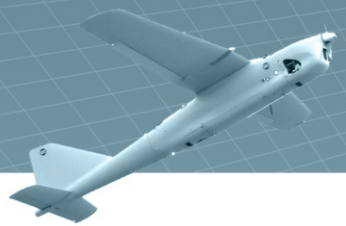
<map>



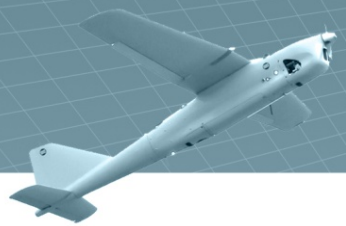
<unordered_set>



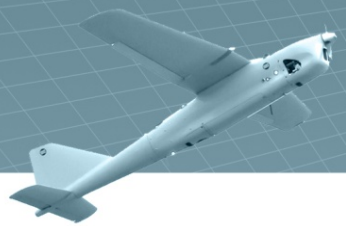
<unordered_map>



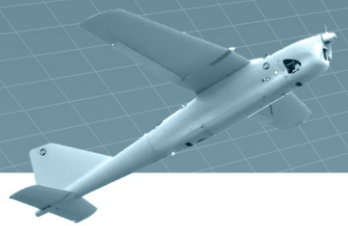
Алгоритмы



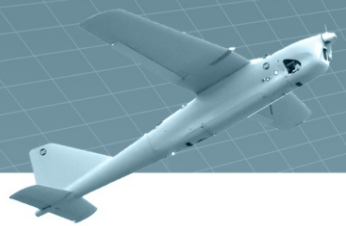
Итераторы



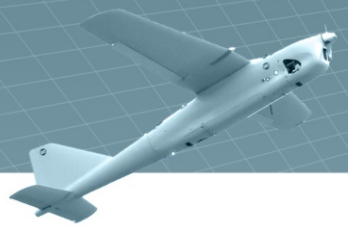
Ввод/вывод



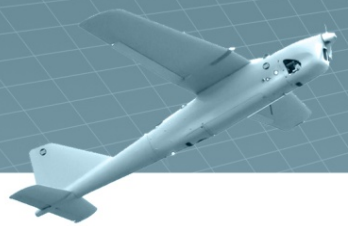
Локализация



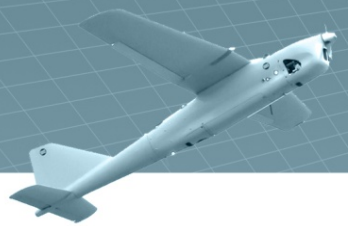
Регулярнве выражения



Локализация



Атомарные операции



Потоки