Лекция

Boost

part 1

what is Boost?

full name: Boost C++ Libraries

http://boost.org/

"The Boost C++ Libraries are a collection of free libraries that extend the functionality of C++"

»Wikipedia

Help

- https://www.boost.org/doc
- https://theboostcpplibraries.com/
- https://stackoverflow.com/questions

Content

- Boost.Test
- Boost.Regex
- Boost.MetaStateMachine
- Boost.TypeTraits
- Boost.Filesystem
- Boost.PropertyTree

Boost.Test

- “A **program** that has **not** been **tested** does **not work**”
- “A systematic way to search for errors”
- “Test early and often”

Bjarne Stroustrup

Boost.Test

- Без написания *модульных тестов* нельзя достигнуть *приемлемого качества* в достаточно сложной системе
- *Повысить скорость разработки* за счет экономии времени на отладке
- *Критично взглянуть на интерфейсы*, выявить несостыковки, обеспечить простоту и логичность
- Если *писать и запускать тесты сложно*, то *покрытие тестами* будет слабое

  - https://github.com/google/googletest
  - https://github.com/unittest-cpp/unittest-cpp
  - https://www.boost.org/doc/libs/1_66_0/libs/test/doc

Boost.Test
организация юнит-теста

- **Юнит-тест** (*модульный тест*) – *отдельное приложение*, при запуске которого происходит выполнение набора тестов и возвращается результат, из которого становится очевидно выполнены ли все тесты без ошибок (txt/xml)
- Из выводимых сообщений можно узнать в каком месте теста произошло нарушение условий
- **Юнит-тест** может состоять из
  - *нескольких файлов с кодом* – физическая организация,
  - *нескольких наборов* (**suite**) *тестов* (**case**) – логическая организация
- **Test case** – несколько утверждений, нарушение которых означает ошибку
- например, *модульные тесты библиотеки* должны содержать несколько наборов тестов, *по набору* на каждый *тестируемый класс*, и в наборе *по тесту* на каждый *тестируемый метод*

boost::test

простой класс, единственная задача которого – деление и умножение

```
1.  #include <stdexcept>
2.  using namespace std;

3.  class Calculator {
4.      int Value_;
5.  public:
6.      explicit Calculator(int value) : Value_(value) {}
7.      void Divide(int value) {
8.          if (value == 0)
9.              throw std::invalid_argument("Деление на ноль!");
10.         Value_ /= value;
11.     }
12.     void Multiply(int value) { Value_ *= value; }
13.     int Result() const { return Value_; }
14. };
```

boost::test

фреймворк

```
1.  #include "calculator.h"
2.  #include <boost/test/unit_test.hpp>

3.  #define BOOST_TEST_MODULE testCalculator

4.  BOOST_AUTO_TEST_CASE(testCalculator)
5.  {
6.      Calculator calculator(12);
7.      BOOST_CHECK_EQUAL(calculator.Result(), 12);
8.      calculator.Divide(3);
9.      BOOST_CHECK_EQUAL(calculator.Result(), 4);
10.     calculator.Divide(2);
11.     BOOST_CHECK_EQUAL(calculator.Result(), 2);
12.     calculator.Multiply(2);
13.     BOOST_CHECK_EQUAL(calculator.Result(), 4);
14.     calculator.Multiply(3);
15.     BOOST_CHECK_EQUAL(calculator.Result(), 12);
16. }
```

Boost.Test
Типы проверок

- **BOOST_AUTO_TEST_SUITE** - определения набора тестов
- **BOOST_CHECK(условие)**
  Простейшая проверка истинно условие или нет
- **BOOST_CHECK_EQUAL(val _1, val_2)**
  Проверка на равенство двух значений
- **BOOST_CHECK_CLOSE(val_1, val_2, точность)**
  Проверка на равенство чисел с плавающей точкой
- **BOOST_CHECK_BITWISE_EQUAL(val_1, val_2)**
  Проверит два значения побитово и сообщит в каком месте биты отличаются
- **BOOST_CHECK_EQUAL_COLLECTIONS(begin1, end1, begin2, end2)**
  Проверка двух последовательностей (контейнеров) на равенства
- **BOOST_CHECK_THROW(инструкция, исключение)**
  Проверка, что при выполнении инструкции будет выброшено указанное исключение.
- **BOOST_CHECK_NO_THROW(инструкция)**
  Здесь наоборот проверяется, что при выполнении инструкции исключений выброшено не будет.

boost::test

фреймворк – пусть будет по тесту на каждый метод

```
1.  #include "calculator.h"
2.  #include <boost/test/unit_test.hpp>
3.  #define BOOST_TEST_MODULE testCalculator

4.  BOOST_AUTO_TEST_CASE(testCalculator) {
5.      Calculator calculator(12);
6.      BOOST_CHECK_EQUAL(calculator.Result(), 12);
7.  }

8.  BOOST_AUTO_TEST_CASE(testCalculatorDivide) {
9.      Calculator calculator(12);
10.     calculator.Divide(3);
11.     BOOST_CHECK_EQUAL(calculator.Result(), 4);
12.     calculator.Divide(2);
13.     BOOST_CHECK_EQUAL(calculator.Result(), 2);
14. }

15. BOOST_AUTO_TEST_CASE(testCalculatorMultiply) {
16.     Calculator calculator(12);
17.     calculator.Multiply(2);
18.     BOOST_CHECK_EQUAL(calculator.Result(), 24);
19.     calculator.Multiply(3);
20.     BOOST_CHECK_EQUAL(calculator.Result(), 72);
21. }
```

boost::test

fixture – конструирование и настройка тестовых объектов

```
1.  #include "calculator.h"
2.  #include <boost/test/unit_test.hpp>
3.  #define BOOST_TEST_MODULE testCalculator
4.  struct Fixture {
5.      Fixture() : calculator(12) { /* Здесь тестовый объект можно настроить */ }
6.      ~Fixture() { /* А здесь корректно завершить с ним работу */ }
7.      Calculator calculator; // тестовый объект
8.  };

9.  BOOST_FIXTURE_TEST_CASE(testCalculator, Fixture) {
10.     BOOST_CHECK_EQUAL(calculator.Result(), 12);
11. }
12. BOOST_FIXTURE_TEST_CASE(testCalculatorDivide, Fixture) {
13.     calculator.Divide(3);
14.     BOOST_CHECK_EQUAL(calculator.Result(), 4);
15.     calculator.Divide(2);
16.     BOOST_CHECK_EQUAL(calculator.Result(), 2);
17. }
18. BOOST_FIXTURE_TEST_CASE(testCalculatorMultiply, Fixture) {
19.     calculator.Multiply(2);
20.     BOOST_CHECK_EQUAL(calculator.Result(), 24);
21.     calculator.Multiply(3);
22.     BOOST_CHECK_EQUAL(calculator.Result(), 72);
23. }
```

Boost.Test

логическая организация

```
1.  #include "calculator.h"
2.  #include <boost/test/unit_test.hpp>
3.  #define BOOST_TEST_MODULE testCalculator
4.  BOOST_AUTO_TEST_SUITE(testSuiteCalculator) // Начало набора тестов

5.  struct Fixture {
6.      // ...
7.  };

8.  BOOST_FIXTURE_TEST_CASE(testCalculator, Fixture) {
9.      // ...
10. }

11. BOOST_FIXTURE_TEST_CASE(testCalculatorDivide, Fixture) {
12.     // ...
13. }

14. BOOST_FIXTURE_TEST_CASE(testCalculatorMultiply, Fixture) {
15.     // ...
16. }

17. BOOST_AUTO_TEST_SUITE_END() // Конец набора тестов
```

Boost.Test

тестирование закрытых методов класса

```
1.  namespace testSuiteCalculator // имя набора тестов
2.  {
3.  // А это имена конкретных тестов
4.  struct testCalculator;
5.  struct testCalculatorDivide;
6.  struct testCalculatorMultiply;
7.  }

8.  class Calculator
9.  {
10.     friend struct ::testSuiteCalculator::testCalculator;
11.     friend struct ::testSuiteCalculator::testCalculatorDivide;
12.     friend struct ::testSuiteCalculator::testCalculatorMultiply;

13. public:
14.     //...
15. };
```

Boost::test
Простая сборка и добавление тестов в проект

- Создается директория **test** в корне проекта
- В этой директории создаются cpp-файлы с тестами (не забываем, что один из них должен содержать определение **BOOST_TEST_MODULE**)

Boost.Test
интеграция сборки CMake & CTest

CMakeLists.txt

```
1. set (TESTS_SOURCES ../tests/файлы_с_тестами.cpp)
2. find_package (Boost COMPONENTS unit_test_framework REQUIRED)
   include_directories(${Boost_INCLUDE_DIRS})
3. set (TEST test_${PROJECT})
4. add_executable (${TEST} ${TESTS_SOURCES})
5. target_link_libraries (${TEST} ${PROJECT} ${Boost_LIBRARIES})
6. enable_testing ()
7. add_test (${TEST} ${TEST})
```

тесты можно запускать выполнив

```
1. make test
```

Boost.Test
Интеграция в сборку CDash

Scan Build *1 build* [view timeline]

		Update	Configure		Build		
Site	Build Name	Revision	Error	Warn	Error	Warn	Start Time ▼
elysium-linux.kitware	Linux-clang-scanbuild	0920c1	0	0	0	0	13 hours ago

Nightly Expected *139 of 148 builds* [view timeline]

		Update	Configure		Build		Test			
Site	Build Name	Revision	Error	Warn	Error	Warn	Not Run	Fail ▼	Pass	Start Time ▼
dash3win7.kitware	vs14-64	0920c1	0	0	0	0	0	4^{+3}_{-4}	507^{+4}_{-3}	12 hours ago
vesper.kitware	vs12-64-ide-intl	0920c1	0	0	0	0	0	3^{+3}_{-3}	415^{+3}_{-3}	11 hours ago
hythloth.kitware	Linux-bullseye-cov	0920c1	0	0	0	0	0	2^{+2}_{-1}	548^{+1}_{-2}	14 hours ago
trinsic.kitware	vs14-64-ninja	0920c1	0	0	0	0	0	2^{+2}	535_{-2}	14 hours ago
hythloth.kitware	Linux-gnu	0920c1	0	0	0	0	0	1^{+1}	660	10 hours ago
trinsic.kitware	icl-64-ninja	0920c1	0	0	0	0	0	1	418	10 hours ago
unstable11s.opencsw.org	Solaris-11-sparc_SunStudio12	0920c1	0	0	0	0	0	1^{+1}	425_{-1}	14 hours ago
unstable11s.opencsw.org	Solaris-11-sparc_SolStudio12u2	0920c1	0	0	0	0	0	1^{+1}	425_{-1}	14 hours ago
unstable11s.opencsw.org	Solaris-11-sparc_SolStudio12u3	0920c1	0	0	0	0	0	1^{+1}	425_{-1}	14 hours ago
unstable11s.opencsw.org	Solaris-11-sparc_SolStudio12u4	0920c1	0	0	0	0	0	1^{+1}	425_{-1}	14 hours ago

boost::regexp → **std::regexp**

- Экземпляры класса **std::regex** определяют *регулярные выражения*

- **std::regex_search** используется для поиска
- **std::regex_replace** используется для операции «*найти и заменить*», вернёт строку после выполнения замены

- Получают на вход регулярное выражение и строку, возвращают найденные результаты в виде экземпляров шаблона **std::match_results**

std::regexp

удобно использовать с «сырыми» строками (можно с обычными)

```
1.  #include <regex>
2.  #include <string>
3.  #include <iostream>

4.  int main() {
5.      const char *reg_exp = R"([ ,.\t\n;:])";  // символы-разделители

6.      std::regex rgx(reg_exp);
7.      std::smatch match;

8.      std::string target{ "\tMoscow State University - in-Tashkent;;" };

9.      while (std::regex_search(target, match, rgx))
10.     {
11.         std::cout << '"' << match.str() << '"' << std::endl;
12.         target = match.suffix();
13.     }
14.     return 0;
15. }
```

regexp

Якоря	
^	Начало строки +
\A	Начало текста +
$	Конец строки +
\Z	Конец текста +
\b	Граница слова +
\B	Не граница слова +
\<	Начало слова
\>	Конец слова

Символьные классы	
\c	Управляющий символ
\s	Пробел
\S	Не пробел
\d	Цифра
\D	Не цифра
\w	Слово
\W	Не слово
\xhh	Шестнадцатиричный символ hh
\Oxxx	Восьмиричный символ xxx

Специальные символы	
\	Экранирующий символ +
\n	Новая строка +
\r	Возврат каретки +
\t	Табуляция +
\v	Вертикальная табуляция +
\f	Новая страница +
\a	Звуковой сигнал
[\b]	Возврат на один символ
\e	Escape-символ
\N{name}	Именованый символ

Кванторы	
*	0 или больше +
*?	0 или больше, нежадный +
+	1 или больше +
+?	1 или больше, нежадный +
?	0 или 1 +
??	0 или 1, нежадный +
{3}	Ровно 3 +
{3,}	3 или больше +
{3,5}	3, 4 или 5 +
{3,5}?	3, 4 или 5, нежадный +

Образцы шаблонов	
([A-Za-z0-9-]+)	Буквы, числа и знаки переноса
(\d{1,2}\/\d{1,2}\/\d{4})	Дата (напр., 21/3/2006)
([^\s]+(?=\.(jpg|gif|png))\.\2)	Имя файла jpg, gif или png
(^[1-9]{1}$\|^[1-4]{1}[0-9]{1}$|^50$)	Любое число от 1 до 50 включительно
(#?([A-Fa-f0-9]){3}(([A-Fa-f0-9]){3})?)	Шестнадцатиричный код цвета
((?=.*\d)(?=.*[a-z])(?=.*[A-Z]).{8,15})	От 8 до 15 символов с минимум одной цифрой, одной заглавной и одной строчной буквой (полезно для паролей).
(\w+@[a-zA-Z_]+?\.[a-zA-Z]{2,6})	Адрес email
(\<(/?[^\>]+)\>)	HTML теги

Упражнение 1,2

1. Вывести число раз, встречается какое-то слово, в строке. Слово и строку принимаем через аргументы командой строки.

Пример
строка: **foo bar (foo) bar foo-bar foo_bar foo'bar bar-foo bar, foo.**

слово: **foo**

ответ: **7**

2. Создать регулярное выражение, которое матчит строки типа

<IP> -- [<date>:<time>]

Пример
66.249.64.13 - - [18/Sep/2004:11:07:48]

Упражнение 3

Match all of these…

✓afoot
✓catfoot
✓dogfoot
✓fanfoot
✓foody
✓foolery
✓foolish
✓fooster
✓footage
✓foothot
✓footle
✓footpad
✓footway
✓hotfoot
✓jawfoot
✓mafoo
✓nonfood
✓padfoot
✓prefool
✓sfoot
✓unfool

and none of these…

✗Atlas
✗Aymoro
✗Iberic
✗Mahran
✗Ormazd
✗Silipan
✗altared
✗chandoo
✗crenel
✗crooked
✗fardo
✗folksy
✗forest
✗hebamic
✗idgah
✗manlike
✗marly
✗palazzi
✗sixfold
✗tarrock
✗unfold

Упражнение 4

Match all of these…

✓xxxxxxx - xx = xxxxx
✓xxxx - x = xxx
✓xxxxxxxx - xxxxx = xxx
✓xx - x = x
✓xxxxxxxxxxx - xxxxx = xxxxxx
✓xxxxx - xx = xxx
✓xxxxxxxx - xx = xxxxxx
✓xxxxxxxxxxxxxxx - xx = xxxxxxxxxxxxx
✓xxxxxxxxxx - xxxxxxxx = xx
✓xxxxxxx - xx = xxxxx
✓xxxxxxxx - xxxxxx = xx
✓xxxxxxxxxxxxx - xxxxxx = xxxxxxx
✓xxx - x = xx
✓xxxxxx - xxxxx = x
✓xxxxxxxxxxxxxx - xxx = xxxxxxxxxxx
✓xxxxxxxxx - xxx = xxxxxx
✓xxxxxxxxxxxxxxxx - xxxxxxxx = xxxxxxxx
✓xxxxx - x = xxxx
✓xxxxxxxxxx - xxxx = xxxxxxx
✓xxxxxxxxxx - xxxxx = xxxxx

and none of these…

✗xxxx - xxx = xx
✗xxxxxxxxxxxxx - xxx = x
✗xxxxxxx - xxxxxxxx = xx
✗xxxxxxx - xxxxx = xxxxxx
✗xx - x = xxxxxx
✗xxxxxxxxx - xxxxxxx = xxxxxxxxxxxxxxx
✗xxxxxxxxxx - xxxxxxxxxxx = xxx
✗x - xx = xxx
✗xxxxxxxxxxxxx - x = xxxx
✗xxx - xxxxxxxxx = xxxxxxxxx
✗xxxxxxxx - x = xxxx
✗xxxxxx - xxxxxxx = xxxxxxxx
✗xxxxxx - xxxx = xxxxxxxxxx
✗xxxxx - xxxxx = xxxxx
✗xxxx - xxxxx = xx
✗xxxxx - xxxxxxxx = xxxxxxxx
✗x - xxxxxxxxxxxxxxx = xxxxxx
✗xxxxx - xxxxxxxxxxxxx = xx
✗xxxxxxxxxxxx - xxxxxxxxxxxx = x
✗xxxxxx - xx = xxx

regexp

stackoverflow

Search...

Home

PUBLIC

Stack Overflow

Tags

Users

Jobs

Teams
Q&A for work

Learn More

RegEx match open tags except XHTML self-contained tags

1324

6179

I need to match all of these opening tags:

```
<p>
<a href="foo">
```

But not these:

```
<br />
<hr class="foo" />
```

I came up with this and wanted to make sure I've got it right. I am only capturing the `a-z`.

```
<([a-z]+) *[^/]*?>
```

I believe it says:

- Find a less-than, then
- Find (and capture) a-z one or more times, then
- Find zero or more spaces, then
- Find any character zero or more times, greedy, except `/`, then
- Find a greater-than

Do I have that right? And more importantly, what do you think?

html regex xhtml

share

edited May 26 '12 at 20:37

community wiki
11 revs, 7 users 58%
Jeff

locked by Robert Harvey ♦ Jun 7 '12 at 19:41

This post has been locked due to the high amount of off-topic comments generated. For extended discussions, please use chat.

Read more about locked posts here.

4421

You can't parse [X]HTML with regex. Because HTML can't be parsed by regex. Regex is not a tool that can be used to correctly parse HTML. As I have answered in HTML-and-regex questions here so many times before, the use of regex will not allow you to consume HTML. Regular expressions are a tool that is insufficiently sophisticated to understand the constructs employed by HTML. HTML is not a regular language and hence cannot be parsed by regular expressions. Regex queries are not equipped to break down HTML into its meaningful parts. so many times but it is not getting to me. Even enhanced irregular regular expressions as used by Perl are not up to the task of parsing HTML. You will never make me crack. HTML is a language of sufficient complexity that it cannot be parsed by regular expressions. Even Jon Skeet cannot parse HTML using regular expressions. Every time you attempt to parse HTML with regular expressions, the unholy child weeps the blood of virgins, and Russian hackers pwn your webapp. Parsing HTML with regex summons tainted souls into the realm of the living. HTML and regex go together like love, marriage, and ritual infanticide. The cannot hold it is too late. The force of regex and HTML together in the same conceptual space will destroy your mind like so much watery putty. If you parse HTML with regex you are giving in to Them and their blasphemous ways which doom us all to inhuman toil for the One whose Name cannot be expressed in the Basic Multilingual Plane, he comes. HTML-plus-regexp will liquify the nerves of the sentient whilst you observe, your psyche withering in the onslaught of horror. Regex-based HTML parsers are the cancer that is killing StackOverflow *it is too late it is too late we cannot be saved* the trangession of a child ensures regex will consume all living tissue (except for HTML which it cannot, as previously prophesied) *dear lord help us how can anyone survive this scourge* using regex to parse HTML has doomed humanity to an eternity of dread torture and security holes *using regex* as a tool to process HTML establishes a breach *between this world* and the dread realm of corrupt entities (like SGML entities, but *more corrupt) a mere glimpse* of the world of reg**ex parsers for HTML will ins**tantly transport a *programmer's consciousness* into a world of ceaseless screaming, he comes, ~~the pestilent slithy~~ regex-infection will **devour your HT**ML parser, application and existence for all time like Visual Basic only worse *he comes he comes do not fight* h**e comes, h**is unholy radiance *destroying all enlightenment, HTML tags* ***leaking from your eyes like liq****uid pain, the song of regular expression parsing* will *extinguish the voices of mortal man from the sphere I can see it can you see it it is beautiful* the final snuf fing *of the lies of Man* ***ALL IS LOST ALL IS L***OST the *pony he comes* he comes ~~he comes~~ **the ich**or permeates *all* MY FACE *MY FACE oh god no* **NO NOOOO N**O stop *the anguish* are **not real ZALGO IS TONY THE PONY HE COMES**

Have you tried using an XML parser instead?

Moderator's Note

This post is locked to prevent inappropriate edits to its content. The post looks exactly as it is supposed to look - there are no problems with its content. Please do not flag it for our attention.

Boost.MetaStateMachine

- **State machines** describe objects through their **states**. They describe what states exist and what transitions between states are possible.
- This lib provides **three** different ways to define state machines.
- **Basic front-end** (expects function pointers) or
  Function front-end (function objects) allow you define state machines in the **conventional way**:
  - create classes,
  - derive them from other classes provided by Boost.MetaStateMachine,
  - define required member variables,
  - write the required C++ code yourself.
- **eUML front-end** (macros) is
  - based on a domain-specific language
  - reusing definitions of a UML state machine
    (can copy definitions from a UML behavior diagram to C++ code).
- You don’t need to work directly with many of the classes provided by **Boost.MetaStateMachine**, just need to know which macros to use, can’t forget to derive your state machine from a class.

boost::msm

```
1.  #include <boost/msm/front/euml/euml.hpp>
2.  #include <boost/msm/front/euml/state_grammar.hpp>
3.  #include <boost/msm/back/state_machine.hpp>
4.  #include <iostream>

5.  namespace msm = boost::msm;
6.  using namespace boost::msm::front::euml;

7.  BOOST_MSM_EUML_STATE((), Off);
8.  BOOST_MSM_EUML_STATE((), On);
9.  BOOST_MSM_EUML_EVENT(press);
10. BOOST_MSM_EUML_TRANSITION_TABLE((((Off + press) == On),((On + press) == Off)),
11.                                 light_transition_table);
12. BOOST_MSM_EUML_DECLARE_STATE_MACHINE((light_transition_table, init_ << Off),
13.                                      light_state_machine);

14. int main() {
15.     msm::back::state_machine<light_state_machine> light;
16.     std::cout << *light.current_state() << '\n';
17.     light.process_event(press);
18.     std::cout << *light.current_state() << '\n';
19.     light.process_event(press);
20.     std::cout << *light.current_state() << '\n';
21.     return 0;
22. }
```

boost::type_traits → **std::type_traits**

разнообразные свойства типов

1. is_const
2. is_standard_layout
3. is_pod
4. is_literal_type
5. is_polymorphic
6. is_abstract
7. is_constructible
8. is_trivially_constructible
9. is_nothrow_constructible
10. is_same
11. is_base_of
12. result_of
13. is_integral
14. is_floating_point
15. is_class
16. is_function
17. is_member_function_pointer
18. is_arithmetic
19. is_reference

std::type_traits

как получить специфичные для типа значения?

```
1.  template <typename C>
2.  auto findMax(const C &container) -> typename C::value_type
3.  {
4.      auto _max = std::numeric_limits<C::value_type>::min();

5.      for (const auto &item : container)
6.          if (_max < item)
7.              _max = item;
8.      return _max;
9.  }
```

std::type_traits

хотим реализовать изменение порядка битов в памяти для POD типов

```
1.  template <typename T>
2.  T byte_inplace_swap(T value) {
3.      std::byte* bytes = reinterpret_cast<std::byte*>(&value);
4.      size_t n = sizeof(T);
5.      for (size_t i = 0; i < (n / 2); ++i)
6.          std::swap(bytes[i], bytes[n - 1 - i]);
7.      return value;
8.  }
```

std::type_traits

допустим, что эта операция допустима не для любого POD типа

```
1.  template <typename T>
2.  T byte_inplace_swap(T value) {
3.      std::byte* bytes = reinterpret_cast<std::byte*>(&value);
4.      size_t n = sizeof(T);
5.      for (size_t i = 0; i < (n / 2); ++i)
6.          std::swap(bytes[i], bytes[n - 1 - i]);
7.      return value;
8.  }
9.  template <>
10. double byte_inplace_swap(double value) {
11.   assert(false && "Illegal to swap doubles");
12.   return value;
13. }
14. template <>
15. char byte_inplace_swap(char value) {
16.   assert(false && "Illegal to swap chars");
17.   return value;
18. }
```

std::type_traits

напишем специализированные флаги, чтобы определить допустимые типы

```
1.  template <typename T>
2.  struct is_swapable { static const bool value = false; };

3.  template <>
4.  struct is_swapable<unsigned short> { static const bool value = true; };

5.  template <>
6.  struct is_swapable<short> { static const bool value = true; };
7.
8.  template <>
9.  struct is_swapable<unsigned long> { static const bool value = true; };
10.
11. template <>
12. struct is_swapable<long> { static const bool value = true; };
13.
14. template <>
15. struct is_swapable<unsigned long long> { static const bool value = true; };
16.
17. template <>
18. struct is_swapable<long long> { static const bool value = true; };
```

std::type_traits

теперь хорошо?

```
1.  #include <type_traits>
2.  template <typename T>
3.  T byte_inplace_swap(T value)
4.  {
5.      assert(is_swapable<T>::value && "Cannot swap this type");
6.      std::byte *bytes = reinterpret_cast<std::byte*>(&value);
7.      size_t n = sizeof(T);
8.      for (size_t i = 0; i < (n / 2); ++i)
9.          std::swap(bytes[i], bytes[n - 1 - i]);
10.     return value;
11. }
```

std::type_traits

лучше

```
1.  #include <type_traits>
2.  template <typename T, typename = std::enable_if<is_swapable<T>::value,T>::type>
3.  T byte_inplace_swap(T value)
4.  {

5.      std::byte *bytes = reinterpret_cast<std::byte*>(&value);
6.      size_t n = sizeof(T);
7.      for (size_t i = 0; i < (n / 2); ++i)
8.          std::swap(bytes[i], bytes[n - 1 - i]);
9.      return value;
10. }
```

std::type_traits

идеально

```
1.  #include <type_traits>

2.  template <typename T>
3.  constexpr bool is_swapable() {
4.      return (std::is_integral<T>::value && (sizeof(T) > 1));
5.  }

6.  template <typename T, typename = std::enable_if<is_swapable<T>(),T>::type>
7.  T byte_inplace_swap(T value)
8.  {
9.      std::byte *bytes = reinterpret_cast<std::byte*>(&value);
10.     size_t n = sizeof(T);
11.     for (size_t i = 0; i < (n / 2); ++i)
12.         std::swap(bytes[i], bytes[n - 1 - i]);
13.     return value;
14. }
```

boost::type_traits

есть некоторые конструкционные блоки, из которых можно создавать флаги

```
1.  template <typename T>
2.  struct is_void : public false_type {};

3.  template <>
4.  struct is_void<void> : public true_type {};


5.  template <typename T>
6.  struct is_pointer : public false_type {};

7.  template <typename T>
8.  struct is_pointer<T*> : public true_type {};
```

Boost.Filesystem

- Делает возможной лёгкую работу с файлами и директориями.
- Предоставляет класс пути **boost::filesystem::path** в файловой системе.
- Пересматривалась несколько раз, текущая версия есть **Boost.Filesystem 3**, начиная с **Boost C++ Libraries 1.46.0**
- Ошибки имеют хорошо читаемые описания:
- terminate called after throwing an instance of 'boost::filesystem::filesystem_error'
  - what(): **boost::filesystem::file_size: Operation not permitted: "."**
  - what(): **boost::filesystem::file_size: No such file or directory: "foo"**
  - what(): **boost::filesystem::status: Permission denied: "/home/jane/foo"** // disk was not ready

boost::filesystem → **std::filesystem**

path

```
1.  #include <clocale>
2.  #include <iostream>
3.  #include <cstdlib>
4.  #include <boost/locale.hpp>
5.  #include <boost/filesystem.hpp>
6.  #include <boost/filesystem/path.hpp>
7.  #include <boost/filesystem/fstream.hpp>

8.  // path: "C:\Users\favorart\source\repos\test\x64\Release\test.exe" \
9.  // file: "test.exe" size is 99328
10. int main(int argc, char **argv) {
11.     namespace fs = boost::filesystem;
12.     std::setlocale(LC_ALL, "English_USA.1251");
13.
14.     fs::path exec(argv[0]);
15.     std::cout << "path: " << exec.make_preferred()
16.               << " file: " << exec.filename()
17.               << " size is " << fs::file_size(exec) << '\n';

18.     fs::path disk{ "C:\\" };
19.     fs::path note{ "C:\\Windows\note.txt" };
20.     fs::path unic{ L"C:\\Boost C++ \u5E93" };
21.     return 0;
22. }
```

std::filesystem

```
1.  // ".." = "C:\Users\favorart\source\repos\test\test\.."
2.  // is a directory false
3.  //     "..\.vs"
4.  //     "..\Debug"
5.  //     "..\test"
6.  //     "..\test.sln"
7.  //     "..\x64"
8.  int main(int argc, char **argv) {
9.     fs::path path(argv[1]);
10.    try {
11.       if (!fs::exists(path)) { std::cout << path << " does not exist\n"; return 1; }

12.       if (fs::is_regular_file(path))
13.          std::cout << path << " size is " << fs::file_size(path) << '\n';
14.       else if (fs::is_directory(path)) {
15.          std::cout << path << " = " << fs::absolute(path) << "\nis a directory "
16.                    << std::boolalpha << path.is_absolute() << "\n";

17.          for (const fs::directory_entry &x : fs::directory_iterator(path))
18.              std::cout << "    " << x.path() << '\n';
19.       }
20.       else std::cout << path << " exists, but what is it?\n";
21.    }
22.    catch (const fs::filesystem_error& ex) { std::cout << ex.what() << '\n'; }
23.    return 0;
24. }
```

std::filesystem

operator / to concatenate paths!

```
1.  int main(int argc, char **argv) {
2.      try {
3.          fs::path complex{ "." };
4.          complex = complex / "complex" / "path" / "to" / "test.txt";
5.          std::cout << complex << " = " << fs::absolute(complex) << '\n';

6.          fs::create_directories(complex.parent_path());
7.          //fs::permissions(complex, fs::perms::remove_perms | fs::perms::others_all);

8.          fs::ofstream myofs{ complex };
9.          myofs << "Hello, world!\n";
10.         myofs.close();

11.         fs::path onemoredir = complex / ".." / ".." / "from";
12.         fs::create_directory(onemoredir);

13.         std::system(("dir " + onemoredir.string()).c_str());

14.         fs::remove_all("complex");
15.     }
16.     catch (const fs::filesystem_error& ex) { std::cout << ex.what() << '\n'; }
17.     return 0;
18. }
```

Boost.Filesystem

```
1.  // ".\complex\path\to\test.txt" = \
2.  // "C:\Users\favorart\source\repos\test\test\.\complex\path\to\test.txt"
3.  //  Том в устройстве C имеет метку SSD_SYSTEM
4.  //  Серийный номер тома: 6A2E-F91F
5.  //
6.  //  Содержимое папки C:\Users\favorart\source\repos\test\test\complex\path\from
7.  //
8.  // 19-Apr-19  14:43    <DIR>          .
9.  // 19-Apr-19  14:43    <DIR>          ..
10. //                0 файлов              0 байт
11. //                2 папок  51 899 994 112 байт свободно



12. // boost::filesystem::directory_iterator
13. // boost::filesystem::recursive_directory_iterator
```

Boost.PropertyTree

Предоставляет возможности работы с древовидной структурой пар ключ-значение. Обрабатывает подразделы, позволяет множественное ветвление

Активно используется для конфигурационных файлов

- **json_parser**
- **ini_parser**

Для отделения подразделов использует символ «точка»

- **get_child**(key) – доступ к подразделу
  - возвращает объект типа, как у родителя
  - т.к. каждый подраздел имеет право иметь свои подразделы, разницы в их типе нет
- **get_value**<type>(key) – получения листового значения нужного типа
- **get_value_or**<type>(key, val) – вернуть значение нужного типа, если в конфиг.файле ключ представлен, иначе вернуть переданное val
- **put**(key, val) – положить значение по ключу
- **put_value**<type>(key, val) – положить подраздел целиком по ключу
- **add_child**(key, child) – добавить подраздел по ключу (даже копией)

boost::property_tree

```
1.  #include <iostream>
2.  #include <boost/property_tree/ptree.hpp>
3.  #include <boost/property_tree/json_parser.hpp>
4.  //#include <boost/property_tree/ini_parser.hpp>
5.  using namespace boost::property_tree;
6.  int main() {
7.      ptree root;
8.      root.put("C:.Windows.System", "20 files");
9.      root.put("C:.Windows.Cursors", "50 files");

10.     boost::optional<std::string> opt = root.get_optional<std::string>("C:");
11.     std::cout << std::boolalpha << opt.get() << '\n';

12.     root.put_child("D:.Program Files", ptree{ "40 files" });
13.     root.put_child("D:.Program Files", ptree{ "50 files" });
14.     root.add_child("D:.Program Files", ptree{ "60 files" });

15.     for (const std::pair<std::string, ptree> &p : root.get_child("D:"))
16.         std::cout << p.second.get_value<std::string>() << '\n';

17.     json_parser::write_json("file.json", root);
18.     ptree copy;
19.     json_parser::read_json("file.json", copy);
20.     std::cout << std::boolalpha << (root == copy) << '\n';
21.     return 0;
22. }
```

boost::property_tree

```
1.  template <typename  T> struct Point {
2.     using value_type = T;
3.     void save(ptree &node) const {
4.        ptree xelem, yelem;
5.        xelem.put_value<Point::value_type>(x); node.push_back(std::make_pair("",xelem));
6.        yelem.put_value<Point::value_type>(y); node.push_back(std::make_pair("",yelem));
7.     }
8.     value_type x, y;
9.  };
10. void write_joints() {
11.    ptree root, joints;
12.    for (const auto &i : { 1, 2, 3, 4 }) {
13.       ptree node;
14.       node.put<unsigned>("joint", 19 + i);
15.       node.put<bool>("show", i % 2);
16.       node.put<float>("frames", 0.4 * i);
17.
18.       Point<double> pt{ i,i }; ptree p; pt.save(p); node.add_child("base", p);

19.       joints.push_back(std::make_pair("", node));
20.    }
21.    root.put<std::string>("pt", "name");
22.    root.add_child("joints", joints);

23.    json_parser::write_json("joints.json", root);
24. }
```

Упражнение 1

- Программа, которая выводит имена всех файлов с расширением **.h/.c/.cpp** во всём компьютере в файл,
- сохраняя иерархию директорий (пропуская директории, где нужных файлов нет).

C:

---VS

--- --projects

--- -- --any

--- -- -- --any.cpp

--- -- -- --any.h

--- -- --fs

--- -- -- --boost.h

Упражнение 2

- Создать программу, которая умеет читать и писать JSON-файлы следующей структуры.
- Если “all” истинно, то при чтении/записи программа должна выводить все имена записанных животных
- Возможность добавления атрибутов животным динамическое по желанию

- { "**animals**": [
- { "**name**": "cat", "legs": 4, "has_tail": true },
- { "**name**": "spider", "legs": 8, "has_tail": false },
  ...
-],
- "**log**": { "all": true }
- }

СПАСИБО ЗА ВНИМАНИЕ!