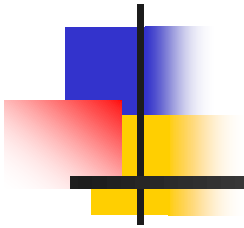


Обзор библиотеки Boost

Попов Владимир

Video Group

CS MSU Graphics & Media Lab

Outline

- Boost
  - Состав библиотеки
  - Лицензия
- Разбор текстовых файлов
- «Умные» указатели

Boost
Состав библиотеки

GRAPHICS & MEDIA LAB
VIDEO GROUP

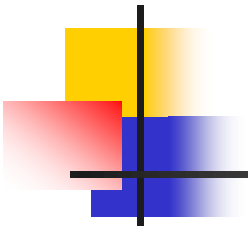
- String and text processing
- Containers
- Iterators
- Algorithms
- Function Objects and higher-order programming
- Generic Programming
- Template Metaprogramming
- Preprocessor Metaprogramming
- Concurrent Programming
- Math and numerics
- Correctness and testing
- Data structures
- Input/Output
- Inter-language support
- **Memory (boost::smart_ptr)**
- **Parsing (boost::spirit)**
- Programming Interfaces
- Miscellaneous

Boost
Лицензия

GRAPHICS & MEDIA LAB
VIDEO GROUP

- http://www.boost.org/users/license.html

- Накладывает ограничения на модификацию библиотеки
- Не накладывает ограничения на код, использующий библиотеку



Outline

- Boost
- Разбор текстовых файлов (boost::spirit)
  - Принцип
  - Пример использования
  - Принцип работы библиотеки
  - Встроенные возможности
- «Умные» указатели

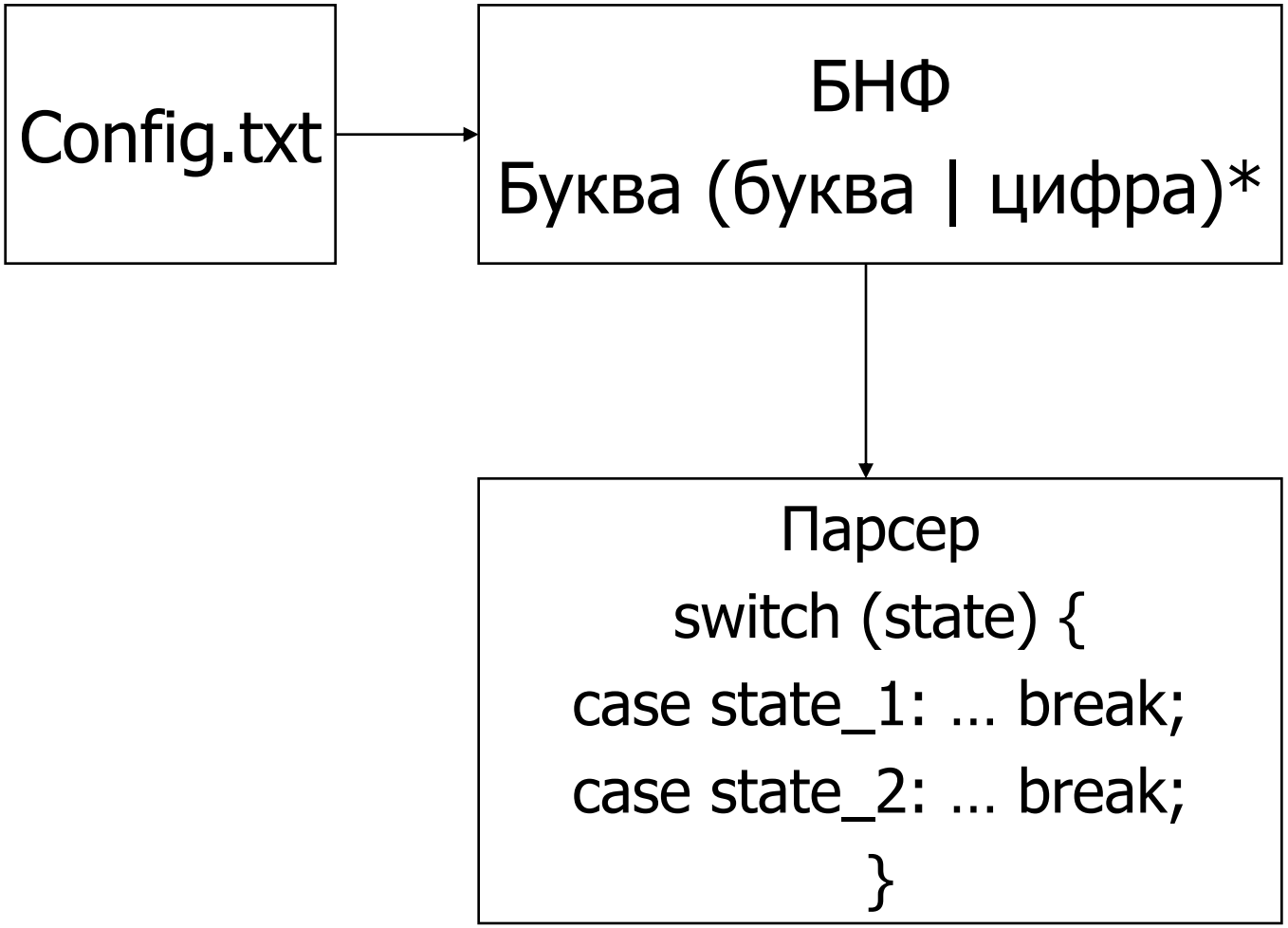
Boost::spirit
Разбор текстового файла

GRAPHICS & MEDIA LAB
VIDEO GROUP

- Удобно хранить параметры программы во внешних файлах
  - Простота изменений
  - Не надо перекомпилировать проект
- Но сложно работать со внешними файлами
  - Сложный формат тяжело читать
  - Если упрощать формат файла, теряется гибкость редактирования данных

Boost::spirit

Разбор текстового файла

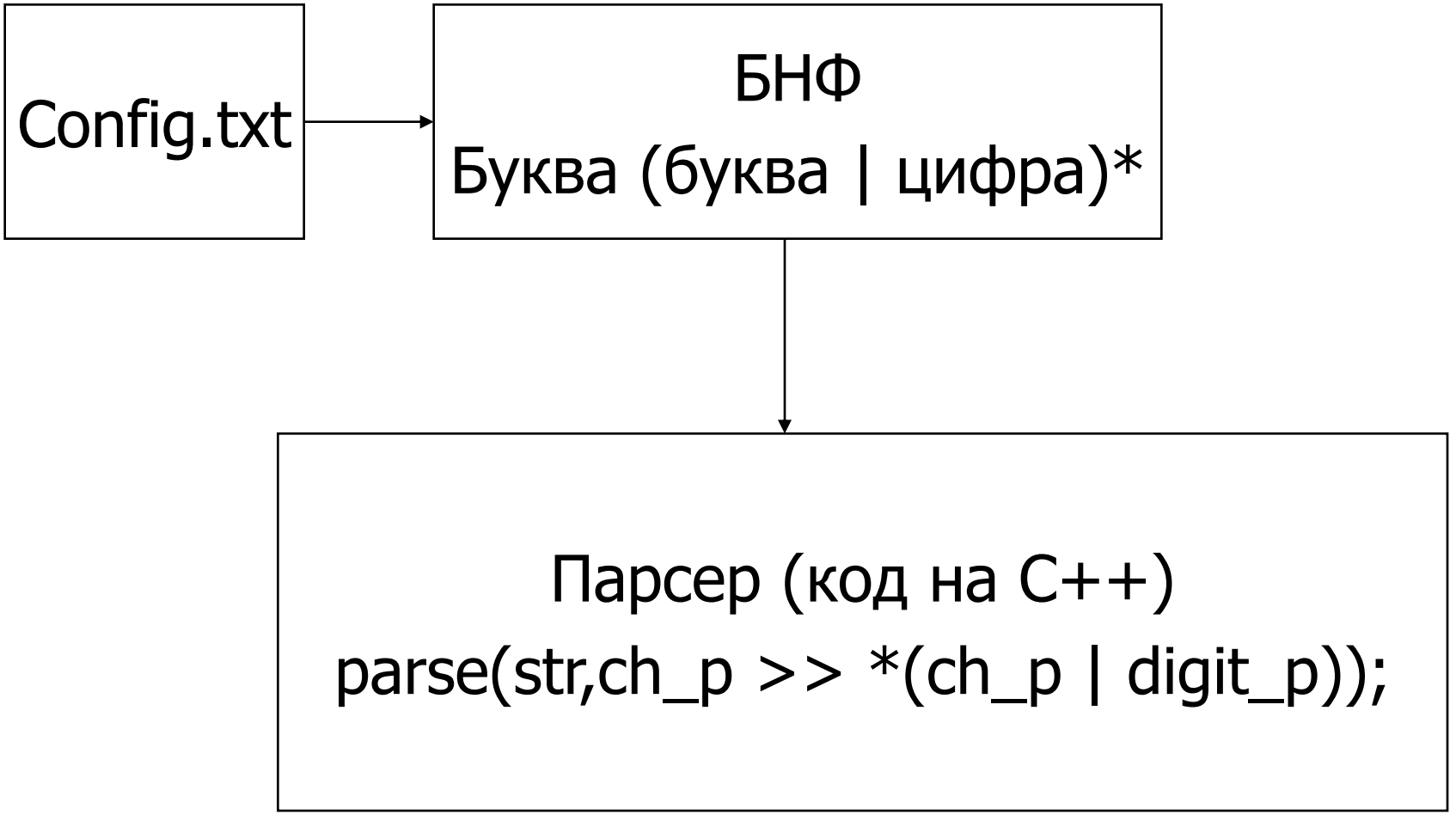


Boost::spirit

Разбор текстового файла. YACC(Bizon)

Boost::spirit

Разбор текстового файла. Boost::spirit



Boost::spirit
Алгоритм использования

GRAPHICS & MEDIA LAB
VIDEO GROUP

1. **Создать правило разбора**
2. Вызвать функцию parse

Boost::spirit
Пример использования

GRAPHICS & MEDIA LAB
VIDEO GROUP

1. Прочитать действительное число

   real_p

2. Прочитать 2 действительных числа

   real_p >> real_p

3. Прочитать произвольное количество действительных чисел

   *real_p

4. Прочитать последовательность действительных чисел через запятую

   real_p >> *(ch_p(',') >> real_p)

Boost::spirit
Алгоритм использования

GRAPHICS & MEDIA LAB
VIDEO GROUP

1. Создать правило разбора
2. **Вызвать функцию parse**

Boost::spirit
Пример использования (2)

GRAPHICS & MEDIA LAB
VIDEO GROUP

5. Вызов функции parse

   parse(str, real_p >> *(',' >> real_p), space_p)

6. Обернем вызов в функцию (законченный пример)

```
bool parse_numbers(char const* str)
{
        return parse(str, real_p >> *(',' >> real_p),
        space_p).full;
}
```

Boost::spirit
Пример использования (3)

GRAPHICS & MEDIA LAB
VIDEO GROUP

7. Сохраним список в массив (семантические действия)

   P – парсер

   a. F – функция. void F (double);
      P[&F] – если разбор прошел успешно, вызвать F для аргумента P

   b. class F
      { public: void operator() (double) { … } }; - функтор
      P[F ()] – вызов функтора

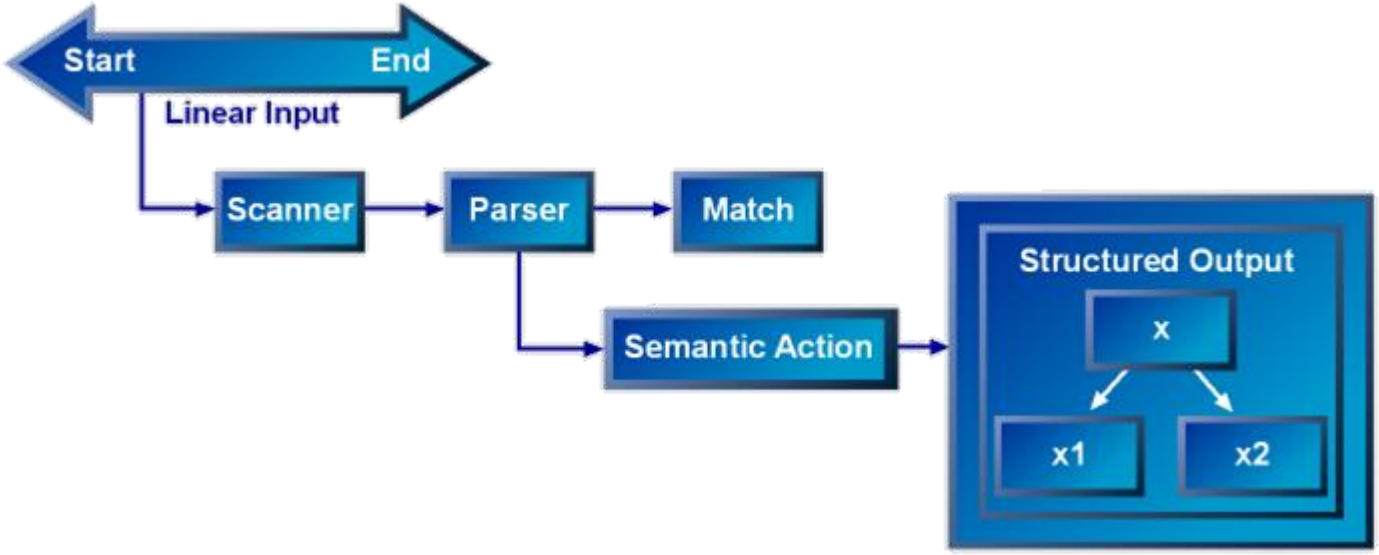
   c. std::vector c;
      push_back_a(c) – встроенный функтор

Boost::spirit
Пример использования (4)

GRAPHICS & MEDIA LAB
VIDEO GROUP

```
bool parse_numbers(const char * str, vector<double>& v)
{
    return parse(str,
        //  Begin grammar
        (
            real_p[push_back_a(v)] >> *(',' >> real_p[push_back_a(v)])
        )
        ,
        //  End grammar
        space_p).full;
}
```

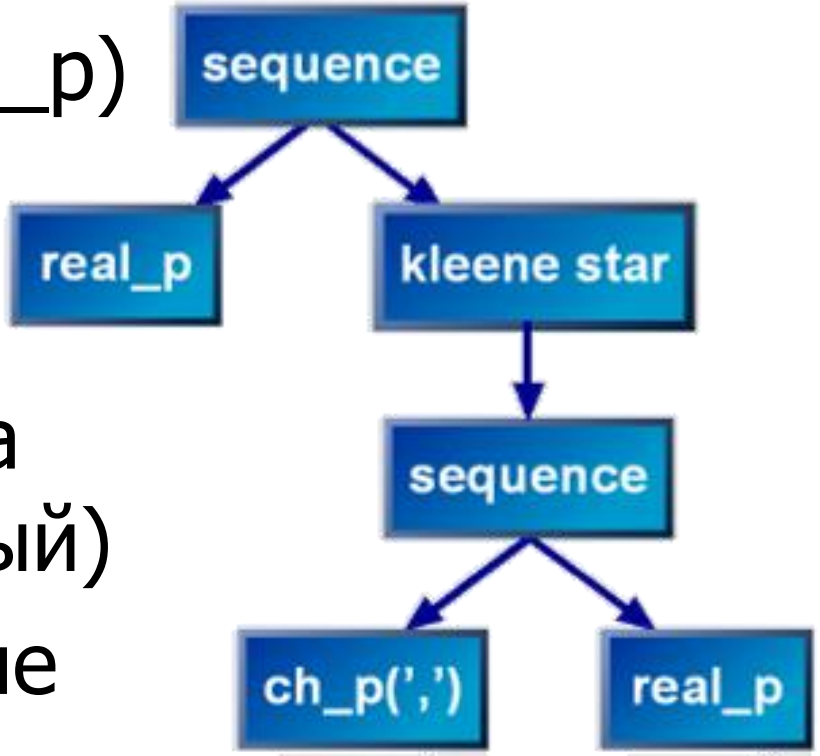
Boost::spirit
Принцип работы



- Парсер выполняет работу по разбору строки
- Большой набор встроенных парсеров
- Возможность написать свой

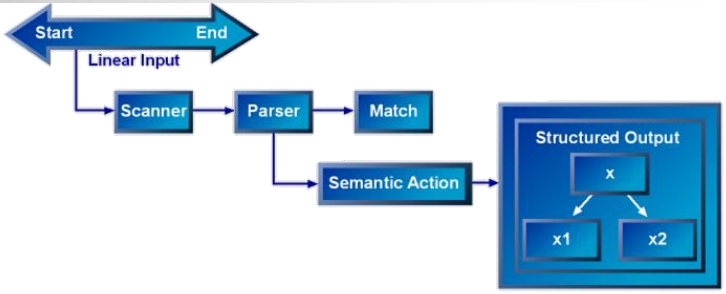
Boost::spirit
Принцип работы. Парсер

- real_p >> *(',' >> real_p)



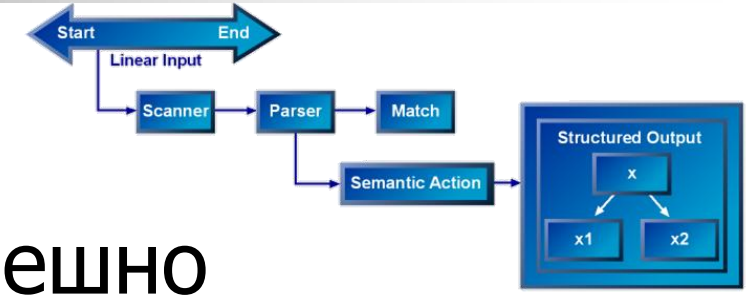
- real_p – объект класса real_parser (встроенный)
- >>, * - перегруженные операторы

Boost::spirit
Принцип работы. Сканер



- Сканер – абстрактная концепция
- Состоит из 2-х итераторов – на текущее положение в потоке и на его конец
- Позволяет настраивать свою работу
  - Удалить пробелы
  - Не различать регистр букв

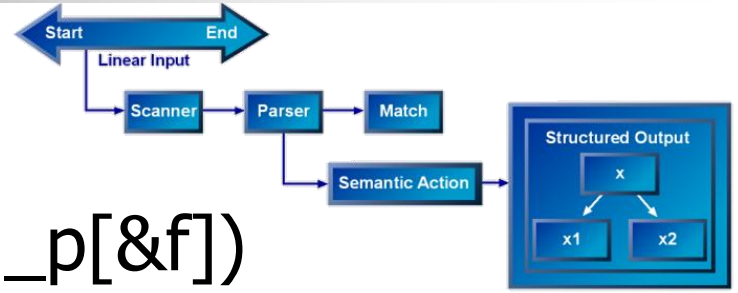
Boost::spirit
Принцип работы. Match



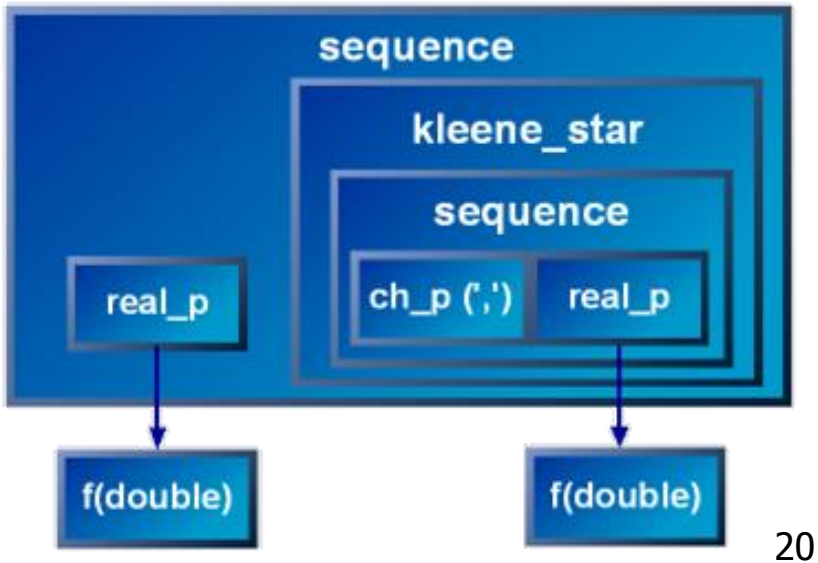
- Задача – сказать, прошел ли разбор успешно
- Может содержать данные в случае успешного разбора (для real_p – действительное число)

Boost::spirit

Принцип работы. Семантические действия



- real_p[&f] >> *(',' >> real_p[&f])
- Разбор успешен – вызывается функция
- Что делать с данными – решает функция
  - Сохранить в массив
  - Сложить все числа



Boost::spirit
Примитивные парсеры

- anychar_p Любой символ
- alnum_p Буквы и цифры
- alpha_p Буквы
- blank_p Пробел и символ табуляции
- cntrl_p Управляющие символы
- digit_p Цифры
- graph_p Видимые символы
- lower_p Прописные буквы
- punct_p Символы-разделители (запятые, точки с запятой…)
- space_p Пробелы, табуляция, перевод строки (\r и \n)
- upper_p Строчные буквы
- xdigit_p Цифры в шестнадцатеричной системе счисления

Boost::spirit
Примитивные парсеры (2)

GRAPHICS & MEDIA LAB
VIDEO GROUP

- ch_p('a') – конкретный символ
- str_p("hello") – конкретная строка
- uint_p – беззнаковое целое число
- int_p – целое число со знаком
- real_p – число с плавающей запятой

Boost::spirit
Операторы(1)

GRAPHICS & MEDIA LAB
VIDEO GROUP

- Теоретико-множественные
  - a | b - Объединение — Подходит a или b
  - a & b - Пересечение — Подходит a и b
  - a – b – разность — Подходит a, но не b. Если a подошло, а для b текст короткий, то операция успешна
  - a ^ b - XOR — Подходит a или b, но не оба сразу
- Последовательность
  - a >> b — Сначала идет a, потом b

Boost::spirit
Операторы(2)

GRAPHICS & MEDIA LAB
VIDEO GROUP

- Циклы
  - *a a встречается 0 или больше раз
  - +a a встречается 1 или больше раз
  - !a a встречается 0 или 1 раз

Пример:

```
rule<> r = ch_p('a')  | ch_p('b') | ch_p('c') | ch_p('d');    // OK
r = ch_p('a')  | 'b' | 'c' | 'd';        // OK
r = 'a' | 'b' | 'c' | 'd';      //неправильно
```

rule – класс, объекты которого хранят парсер

Boost::spirit
Вспомогательные парсеры

- confix_p(start,string,end)
- comment_p(start, end)
- comment_p(start)
- list_p(item,delimeter)

Boost::spirit
Действия

GRAPHICS & MEDIA LAB
VIDEO GROUP

```
parser[&my_function]
parser[my_functor]

class my_functor
{
public:
    template<class IteratorT>
    void operator() const (IteratorT begin, IteratorT end)
    {    …
    }
};
```

Boost::spirit
Встроенные действия

GRAPHICS & MEDIA LAB
VIDEO GROUP

- increment_a(ref)
- decrement_a(ref)
- assign_a(ref)
- assign_a(ref, value)
- push_back_a(ref)
- push_back_a(ref, value)
- clear_a(ref)

- Пример:

```
bool state;
vector<double> c;
double num;
rule<> r = (str_p("state_f")[assign_a(state,false)] |
                    str_p("state_t")[assign_a(state,true)]
) >> int_p[my_action(state)]
 >> real_p[push_back_a(c)] >> real_p[assign_a(num)];
```

Boost::spirit
Функция parse

- parse (str, rule) – символьный уровень
- parse (str, rule, skip) – уровень фраз

- parse (iterator_begin, iterator_end,…)

- Возвращает parse_info.
  - parse_info::hit – прошел ли парсинг успешно
  - parse_info::full – был ли считан весь ввод

Boost::spirit
Пример. Конфиг файл

```
###COMMON
set bitrate=%7
set /A bitrate = %bitrate%*1024*8
video_enc_con.exe h261 -i %1 -n %6 -w %3 -h %4 -f %5 -b %bitrate%
                        %PARAMETERS% %PARAMETERS_FILE% %2
###END
###DECODING
move %1 tmp
ldecod.exe -i tmp -o %2
move tmp %1
###END
###PARAMETERS
"-t @"      ENUM [0,1]          1
###END
###PARAMETERS_FILE
"@"         SOURCE_FILE
"@"         WIDTH
…
```

```
…
"@"         HEIGHT
"@"         FRAMES_NUM
"@"         ENUM     [0,1]        1
"@"         BITRATE_BPS
"45"        QUOTE
"@"         ENUM     [0..3]       2
"@"         ENUM     [0,1]        1
"@"         ENUM     [“esa”,”umh”]  “umh”
"@"         ENUM     [0,1,2]      0
"@"         ENUM     [0,1]        1
"12:11"     QUOTE
###END
```

Boost::spirit
Пример. Чтение конфиг файла

GRAPHICS & MEDIA LAB
VIDEO GROUP

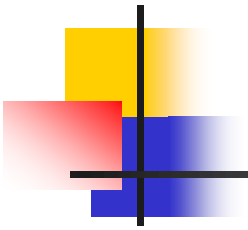
```
typedef rule<phrase_scanner_t> rule_t;

rule_t q_string = confix_p ("\"", (+anychar_p), "\"");
rule_t str_vec = list_p(q_string, ',');
rule_t int_vec = list_p(uint_p,',');
rule_t rule_parameter =
(
 (confix_p ("\"", (+anychar_p), "\"") >>
  (str_p ("TOGGLE")     | str_p ("SOURCE_FILE")| str_p ("WIDTH") | str_p ("HEIGHT") |
   str_p ("FRAMES_NUM") | str_p ("BITRATE_BPS")| str_p ("QUOTE") | str_p ("ENUM") >>
   (
    (confix_p ('[', str_vec, ']') >> q_string) |
    (confix_p ('[', (uint_p >> ".." >> uint_p >> !(',' >> uint_p)), ']') >> uint_p) |
    (confix_p ('[', int_vec, ']') >> uint_p)
   )))
);

rule_t rule_parameters = *(rule_parameter);
rule_t rule_paramfile = *(rule_parameter);

parse_info<> res = parse (pData,
 confix_p ("###COMMON", (*anychar_p), "###END") >>
 confix_p ("###DECODING", (*anychar_p), "###END") >>
 (!confix_p (str_p("###PARAMETERS"), rule_parameters, "###END")) >>
 (!confix_p (str_p("###PARAMETERS_FILE"), rule_paramfile, "###END"))
 ,space_p
);
```

Boost::spirit
Преимущества и недостатки

GRAPHICS & MEDIA LAB
VIDEO GROUP

+ Позволяет быстро сделать парсер относительно простых данных (компилятор C++ писать не рекомендуется)
+ Хорошая читабельность парсера (можно разобрать формат разбираемого текста)

- Сложность отладки
- Быстро увеличивается время компиляции с увеличением числа правил



Outline

- Boost
- Разбор текстовых файлов
- «Умные» указатели
  - Назначение
  - Состав
  - scoped_ptr
  - shared_ptr
  - weak_ptr
  - intrusive_ptr

«Умные» указатели
Назначение

GRAPHICS & MEDIA LAB
VIDEO GROUP

- Хранят указатель на динамический объект
- Отвечают за удаление объекта в нужное время

```
smart_ptr<int> ptr (new int(5));
// delete не нужен
```

- Отвечают за управление объектом, используемым в нескольких местах

```
std::vector<smart_ptr<int> > c;
c.push_back(ptr);
// указатель удалится из c и из функции в
// нужное время автоматически
```

- Полезны при обработке исключений

```
void f () {
        smart_ptr<int> ptr (new int (5));
        throw Exception;
}
```

«Умные» указатели
Состав

- scoped_ptr – один владелец объекта. Нельзя копировать
- scoped_array – один владелец массива. Нельзя копировать
- shared_ptr – несколько владельцев объекта
- shared_array – несколько владельцев массива
- weak_ptr – просмотрщик объекта, который содержится в shared_ptr
- intrusive_ptr – несколько владельцев объекта, внешний подсчет ссылок

«Умные» указатели
scoped_ptr

GRAPHICS & MEDIA LAB
VIDEO GROUP

- Работает быстро и не требует дополнительной памяти
- Удалит объект автоматически в нужное время
- Нельзя копировать и использовать в контейнерах Стандартной Библиотеки

«Умные» указатели
scoped_ptr

GRAPHICS & MEDIA LAB
VIDEO GROUP

```
namespace boost {
  template<class T> class scoped_ptr : noncopyable {
   public:
     typedef T element_type;
     explicit scoped_ptr(T * p = 0); // never throws
     ~scoped_ptr(); // never throws
     void reset(T * p = 0); // never throws
     T & operator*() const; // never throws
     T * operator->() const; // never throws
     T * get() const; // never throws

     operator bool() const; // never throws
     void swap(scoped_ptr & b); // never throws
  };
  template<class T> void swap(scoped_ptr<T> & a, scoped_ptr<T> & b);
// never throws
}
```

«Умные» указатели
scoped_ptr vs auto_ptr

GRAPHICS & MEDIA LAB
VIDEO GROUP

- Не допустить копирование указателя
- Показать будущим разработчикам, что указатель не должен копироваться
- Эквивалентен std::auto_ptr<T> const, но нельзя вызвать reset

«Умные» указатели
scoped_ptr. Пример

GRAPHICS & MEDIA LAB
VIDEO GROUP

```
#include <boost/scoped_ptr.hpp>
#include <iostream>
struct Shoe { ~Shoe() { std::cout << "Buckle my shoe\n"; } };
class MyClass {
    boost::scoped_ptr<int> ptr;
  public:
    MyClass() : ptr(new int) { *ptr = 0; }
    int add_one() { return ++*ptr; }
};
int main()
{
    boost::scoped_ptr<Shoe> x(new Shoe);
    MyClass my_instance;
    std::cout << my_instance.add_one() << '\n';
    std::cout << my_instance.add_one() << '\n';
}
```

Вывод:

```
1
2
Buckle my shoe
```

«Умные» указатели shared_ptr

GRAPHICS & MEDIA LAB
VIDEO GROUP

- Поддерживает копирование указателей
- Использует подсчет ссылок
- Автоматически удаляет объект
- Поддерживает преобразование типов указателей
  shared_ptr<T1> -> shared_ptr<T2>

- Не работает с циклическими ссылками (использовать weak_ptr)

«Умные» указатели shared_ptr. Принцип применения

- Каждое создание объекта должно иметь вид:
  shared_ptr<T> p(new Y);
- Пример:
  - Выполняется new int(2)
  - Выполняется g()
  - В g выбрасывается исключение
  - Результат: указатель не создан,
    утечка памяти

```
void f(shared_ptr<int>, int);
int g();
void ok()
{
    shared_ptr<int> p(new int(2));
    f(p, g());
}
void bad()
{
    f(shared_ptr<int>(new int(2)),
  g());
}
```

«Умные» указатели shared_ptr. Пример


```
struct Foo
{
  Foo( int _x ) : x(_x) {}
  ~Foo() { std::cout << "Destructing a Foo
with x=" << x << "\n"; }
  int x;
  /* ... */
};
typedef boost::shared_ptr<Foo> FooPtr;
struct FooPtrOps
{
  bool operator()( const FooPtr & a, const
FooPtr & b )
    { return a->x > b->x; }
  void operator()( const FooPtr & a )
    { std::cout << a->x << "\n"; }
};
```

«Умные» указатели shared_ptr. Пример

```
int main()
{
  std::vector<FooPtr>          foo_vector;
  std::set<FooPtr,FooPtrOps>   foo_set;
 // NOT multiset!
  FooPtr foo_ptr( new Foo( 2 ) );
  foo_vector.push_back( foo_ptr );
  foo_set.insert( foo_ptr );
  foo_ptr.reset( new Foo( 1 ) );
  foo_vector.push_back( foo_ptr );
  foo_set.insert( foo_ptr );
  foo_ptr.reset( new Foo( 3 ) );
  foo_vector.push_back( foo_ptr );
  foo_set.insert( foo_ptr );
  foo_ptr.reset ( new Foo( 2 ) );
  foo_vector.push_back( foo_ptr );
  foo_set.insert( foo_ptr );
  std::cout << "foo_vector:\n";
  std::for_each( foo_vector.begin(),
foo_vector.end(), FooPtrOps() );

  std::cout << "\nfoo_set:\n";
  std::for_each( foo_set.begin(),
foo_set.end(), FooPtrOps() );
  std::cout << "\n";
  return 0;
}
```

Вывод на экран:

```
foo_vector:
2
1
3
2

foo_set:
3
2
1

Destructing a Foo with x=2
Destructing a Foo with x=1
Destructing a Foo with x=3
Destructing a Foo with x=2
```

«Умные» указатели
weak_ptr

GRAPHICS & MEDIA LAB
VIDEO GROUP

- Хранит слабую ссылку
- Доступ к объекту:
  - Конструктор shared_ptr
  - Функция shared_ptr<T> lock ()
- Устраняет проблему циклических зависимостей

  Внутри объекта содержится ссылка на сам объект

«Умные» указатели weak_ptr. Пример

GRAPHICS & MEDIA LAB
VIDEO GROUP

```
shared_ptr<int> p(new int(5));
weak_ptr<int> q(p);

// some time later

if(shared_ptr<int> r = q.lock())
{
    // use *r
}
```

«Умные» указатели intrusive_ptr

GRAPHICS & MEDIA LAB
VIDEO GROUP

- Внешний подсчет ссылок
  - При копировании вызвает intrusive_ptr_add_ref(T* p), p – указатель
  - При удалении – intrusive_ptr_release(T* p)
- Объект класса intrusive_ptr занимает в памяти столько же места, как и обычный указатель

  sizeof (intrusive_ptr<T>) = sizeof (T*)

«Умные» указатели intrusive_ptr. Пример


```
class base
{
private:
    int use_count_;
public:
    base(): use_count_(0)
    {
    }
    virtual ~base()
    {
    }

    void add_ref()
    {
        ++use_count_;
    }
    void release()
    {
        if(--use_count_ == 0)
         delete this;
    }
};
```

```
inline void intrusive_ptr_add_ref(base * p)
{
    p->add_ref();
}
inline void intrusive_ptr_release(base * p)
{
    p->release();
}
int main ()
{
    intrusive_ptr<base*> p (new base ());
}
```

http://www.boost.org

boost C++ LIBRARIES

"...one of the most highly regarded and expertly designed C++ library projects in the world."
— Herb Sutter and Andrei Alexandrescu, C++ Coding Standards

Welcome to Boost.org!

Boost provides free peer-reviewed portable C++ source libraries.

We emphasize libraries that work well with the C++ Standard Library. Boost libraries are intended to be widely useful, and usable across a broad spectrum of applications. The Boost license encourages both commercial and non-commercial use.

We aim to establish "existing practice" and provide reference implementations so that Boost libraries are suitable for eventual standardization. Ten Boost libraries are already included in the C++ Standards Committee's Library Technical Report (TR1) as a step toward becoming part of a future C++ Standard. More Boost libraries are proposed for the upcoming TR2.

Getting Started

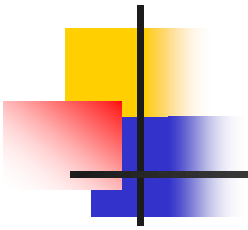
Boost works on almost any modern operating system, including UNIX and Windows variants. Follow the Getting Started Guide to download and install Boost. Popular Linux

Search

Google

- Introduction
- Community
- Development
- Support
- Documentation
- Recent Announcements

Review finished
March 30th, 2008 23:06 GMT



Вопросы

?