Ruby and C bindings

- Ruby is powerful language
- There are many libraries written in C
- Bind them
- Profit

Genrating Makefile

extconf.rb:

```
#!/usr/bin/env ruby

require 'mkmf'
dir_config('rubyc')
create_makefile('rubyc')
```

```
$ ruby extconf.rb
creating Makefile
```

RubyC beginning

rubyc.c:

```
#include "ruby.h"

static VALUE rb_mRubyC;
static VALUE rb_cTestClass;

void Init_rubyc() {
  rb_mRubyC = rb_define_module("RubyC");

  rb_cTestClass = rb_define_class_under(rb_mRubyC,
"TestClass", rb_cObject);
}
```

Just works!

```
$ make
...
...
$ irb
irb(main):001:0> require 'rubyc'
=> true
irb(main):002:0> test_instance = RubyC::TestClass.new
=> #<RubyC::TestClass:0xb7cc0b34>
```

Hello, World!

```
static VALUE say_hello() {
  printf("hello, world!\n");
  return Qnil;
}

rb_define_method(rb_cTestClass, "say_hello", say_hello, 0);
```

```
irb(main):003:0>
test_instance.say_hello
hello, world!
=> nil
```

Exception example

```
static VALUE throw_error()
{ rb_raise(rb_eStandardError, "error");
}

rb_define_method(rb_cTestClass, "throw_error", throw_error, 0);

irb(main):004:0>
test_instance.throw_error
StandardError: error
        from (irb):4:in `throw_error'
        from (irb):4
        from :0
irb(main):005:0>
```

More complicated print

```
static VALUE say(VALUE self, VALUE obj) {
  if (rb_respond_to(obj, rb_intern("to_s"))) {
    VALUE str = rb_funcall(obj, rb_intern("to_s"), 0);
    printf("%s\n", StringValuePtr(str));
  }
  else {
    rb_raise(rb_eRuntimeError, "Unexpected parameter");
  }
  return Qnil;
}

rb_define_method(rb_cTestClass, "say", say, 1);
```

Print anything!

```
irb(main):005:0> test_instance.say 'hello'
hello
=> nil
irb(main):006:0> test_instance.say ['asdfg', 1, 2]
asdfg12
=> nil
irb(main):007:0> test_instance.say :a => :b, :c => :d
cdab
=> nil
```

Remember last to instance variable

```
rb_ivar_set(self, rb_intern("@last"), obj);
```

```
static VALUE last(VALUE self) {
  return rb_ivar_get(self, rb_intern("@last"));
}

rb_define_method(rb_cTestClass, "last", last, 0);
```

Try whether our object has memory

```
irb(main):003:0> a.last
=> nil
irb(main):004:0> a.say "Labas rytas!"
Labas rytas!
=> nil
irb(main):005:0> a.last
=> "Labas rytas!"
```