The Important Art of Thinking

Charles Bailey

28th April 2012

Thinking

Writing software is all about thinking.

Thinking

Writing software is all about thinking.

If you are simply following rules you are putting the computer out of a job.

When?

When should we think?

When?

When should we think?

With hindsight, we can recognize when we should have started thinking.

Better thinking

Experience leads to better thinking.

For the novice

Use `std::endl` to end lines that your program outputs.

For the advanced beginner

`std::endl` is a manipulator.

For the advanced beginner

`std::endl` is a function.

`std::ostream` has a member `operator<<` that takes a pointer to a function taking and returning a reference to a `std::ostream` and executes the passed function passing `*this`.

What is endl?

```
namespace std {
  template <class charT, class traits>
    basic_ostream<charT,traits>& endl(
      basic_ostream<charT,traits>& os);
}
```

Effects: Calls `os.put(os.widen('\n'))`, then `os.flush()`.
Returns: `os`.

What is endl?

```
namespace std {
  template <class charT, class traits>
    basic_ostream<charT,traits>& endl(
      basic_ostream<charT,traits>& os);
}
```

Effects: Calls `os.put(os.widen('\n'))`, then `os.flush()`.
Returns: `os`.

“Hello, world” revisited

```
#include <iostream>
int main() {
    endl(std::cout << "Hello, world!");
}
```

Javascript

```
var g_counter = 0;
function getId() {
    return ++g_counter;
}
```

Javascript has let

```
10 LET a = 10;
20 PRINT "a=", a;
```

Javascript

```
let getId = function() {
    let counter = 0;
    return function() { return ++counter; }
}();
```

Javascript

```
var g_foo = createFoo();
```

Javascript

```
let getFoo = function() {
    let foo;
    return function() {
        if (!foo)
            foo = createFoo();
        return foo;
    };
} ();
```

Large scale Copy & Paste

```
auto li = char(std::lower(c));
```

Large scale Copy & Paste

```
auto li = char(std::lower(c));
```

5.2.3 Explicit type conversion (functional notation)

[...] If the expression list is a single expression, the type conversion expression is equivalent (in definedness, and if defined in meaning) to the corresponding cast expression.

Large scale Copy & Paste

```
auto li = (char)std::lower(c);
```

5.2.3 Explicit type conversion (functional notation)

[...] If the expression list is a single expression, the type conversion expression is equivalent (in definedness, and if defined in meaning) to the corresponding cast expression.

The pointless header

Including <iso646.h> or <ciso646> has no effect. In C++ you can already do this.

```
if (not! good) {
    // ...
}
```

N
L
P

N
L
Pattern

N
Lambda
Pattern

Named Lambda Pattern

What's in a name?

```
int main()
{
    [] () {} ();
}
```

What's in a name?

```
int main()
{
    auto doNothing = [] () {};
    doNothing();
}
```

“Hello, world” re-revisited

```
#include <iostream>
auto Main = [] { endl(std::cout<<"Hello, world!"); };
int main() {
    Main();
}
```

And now a message from LaTeX

```
Error:  Weird page contents
```

NLP in Python

```
>>> addOne = lambda(x): x + 1
>>> print addOne(5)
6
```

NLP in Perl

```
my $subOne = sub { return $_[0] - 1; };
print $subOne->(5) . "\n";
4
```

STOP

STOP
LOOK

STOP
LOOK
THINK