Regular Expressions Redefined in C++

hana.dusikova@avast.com

Regular Expressions in C++

- easy to write
- REs definition usually doesn’t change

- interpreted
- large footprint

```
RegExp<Begin,
Select<Char<'a','b','c'>,String<'x','y','z'>>,
Plus<Anything>,End>
```

```
^(?:[abc]|xyz).+$
```

```
using RE = RegExp<Plus<Char<'l'>>>;

static_assert(RE::match("hello"));
```

```
using Ls = Plus<Char<'l'>>;
using RE = RegExp<Ls>;

static_assert(RE::match("hello"));
```

```
using RE =
 RegExp<Plus<Catch<1,Word>,Opt<White>>>;

RE re;
re.match("hello there");

for (auto str: re.getCatch<1>()) {
  cout << str << "\n";
}
```

```
RegExp<Begin,
Select<Char<'a','b','c'>,String<'x','y','z'>>,
Plus<Anything>,End>
```

```
^(?:[abc]|xyz).+$
```

```
RegExp<Begin,
Select<Char<'a','b','c'>,String<'x','y','z'>>,
Plus<Anything>,End>
```

```
"^(?:[abc]|xyz).+$"
```

```
RegExp<Begin,
Select<Char<'a','b','c'>,String<'x','y','z'>>,
Plus<Anything>,End>
```

```
auto re = "^(?:[abc]|xyz).+$"_pre;
```

```
using RE = RegExp<Begin,
Select<Char<'a','b','c'>,String<'x','y','z'>>,
Plus<Anything>,End>
```

```
auto re = "^(?:[abc]|xyz).+$"_pre;
static_assert(is_same(RE,decltype(re)));
```

```
#include "pregexp.hpp"

int main(int argc, char ** argv) {
 using namespace sre;

 if (argc >= 2 && "^(?:[abc]|xyz).+$"_pre.match(argv[1])) {
  puts("match");
  return 0;
 } else {
  puts("not match");
  return 1;
 }
}
```

```
_main:
0000000100000e70   pushq   %rbp
0000000100000e71   movq    %rsp, %rbp
0000000100000e74   cmpl    $0x2, %edi
0000000100000e77   jl      0x100000f3d
0000000100000e7d   movq    0x8(%rsi), %r9
0000000100000e81   movb    (%r9), %al
0000000100000e84   testb   %al, %al
0000000100000e86   je      0x100000f3d
0000000100000e8c   movzbl  %al, %r8d
0000000100000e90   cmpl    $0x61, %r8d
0000000100000e94   je      0x100000ea0
0000000100000e96   andb    $-0x2, %al
0000000100000e98   movzbl  %al, %eax
0000000100000e9b   cmpl    $0x62, %eax
0000000100000e9e   jne     0x10000
0000000100000ea0   leaq    0x1(%r9
0000000100000ea4   xorl    %edx, %edx
0000000100000ea6   movq    %r9, %rdi
0000000100000ea9   xorl    %r10d, %r10d
0000000100000eac   jmp     0x100000ebf
0000000100000eae   nop
0000000100000eb0   movq    %rdi, %rax
0000000100000eb3   addq    $0x2, %rax
0000000100000eb7   decl    %edx
0000000100000eb9   movq    %rsi, %rdi
0000000100000ebc   movq    %rax, %rsi
0000000100000ebf   testl   %edx, %edx
0000000100000ec1   movb    (%rsi), %al
0000000100000ec3   je      0x100000ed1
0000000100000ec5   testb   %al, %al
0000000100000ec7   movb    $0x1, %cl
0000000100000ec9   je      0x100000ece
0000000100000ecb   movb    %r10b, %cl
0000000100000ece   movb    %cl, %r10b
0000000100000ed1   testb   %al, %al
0000000100000ed3   jne     0x100000eb0
0000000100000ed5   testb   $0x1, %r10b
0000000100000ed9   jne     0x100000f50
0000000100000edb   cmpl    $0x78, %r8d
0000000100000edf   jne     0x100000f3d
0000000100000ee1   movzbl  0x1(%r9), %eax
0000000100000ee6   cmpl    $0x79, %eax
0000000100000ee9   jne
0000000100000eeb   movzbl
0000000100000ef0   cmpl    $0x7a, %eax
0000000100000ef3   jne     0x100000f3d
0000000100000ef5
0000000100000ef9   addq
0000000100000efd   xorl    %edx, %edx
0000000100000eff   xorl    %esi, %esi
0000000100000f01   nopw    %cs:(%rax,%rax)
0000000100000f10   movq    %r9, %rdi
0000000100000f13   testl   %edx, %edx
0000000100000f15   movb    0x2(%rcx), %al
0000000100000f18   je      0x100000f27
0000000100000f1a   testb   %al, %al
0000000100000f1c   movb    $0x1, %r8b
0000000100000f1f   je      0x100000f24
0000000100000f21   movb    %sil, %r8b
0000000100000f24   movb    %r8b, %sil
0000000100000f27   addq    $0x2, %rcx
0000000100000f2b   decl    %edx
0000000100000f2d   testb   %al, %al
0000000100000f2f   movq    %rcx, %r9
0000000100000f32   movq    %rdi, %rcx
0000000100000f35   jne     0x100000f10
0000000100000f37   testb   $0x1, %sil
0000000100000f3b   jne     0x100000f50
## literal pool for: "not match"
0000000100000f3d   leaq    0x44(%rip), %rdi
## symbol stub for: _puts
0000000100000f44   callq   0x100000f60
0000000100000f49   movl    $0x1, %eax
0000000100000f4e   popq    %rbp
0000000100000f4f   retq
## literal pool for: "match"
0000000100000f50   leaq    0x2b(%rip), %rdi
## symbol stub for: _puts
0000000100000f57   callq   0x100000f60
0000000100000f5c   xorl    %eax, %eax
0000000100000f5e   popq    %rbp
0000000100000f5f   retq
```

Only 78 instructions.
No recursion.

cpp.fail/re