Dr Ivan Čukić
KDAB, KDE

(DON'T) USE COROUTINES FOR THIS

INTRO TO
RECURSION
OF HAPPINESS
IN THE ZEN GARDEN
OF CHAINS

ABOUT ME

- KDAB PM, TL, SSE
  Trusted Software Excellence
- Author of “Functional Programming in C++”
  available in English, Chinese, Korean, Russian, Polish
- Trainer / Consultant
- ISO WG21
- KDE developer
- Former university professor

INTRO TO
RECURSION
OF HAPPINESS
IN THE ZEN GARDEN
OF CHAINS

Ivan Čukić kdab.com, cukic.co

PROG C++

Too old to Rock 'n' Roll,
to young to die

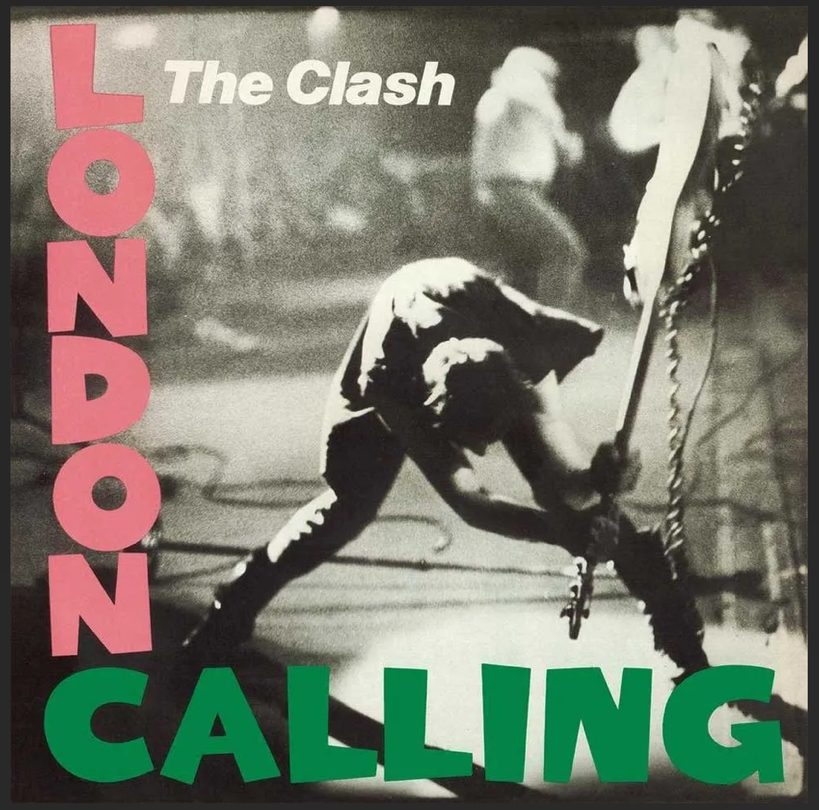
– Ian Anderson

INTRO TO RECURSION OF HAPPINESS IN THE ZEN GARDEN OF CHAINS

Meeting C++ 2025

Ivan Čukić kdab.com, cukic.co

PUNK C++



Berlin Calling

– Meeting C++

INTRO TO RECURSION OF HAPPINESS IN THE ZEN GARDEN OF CHAINS

Meeting C++ 2025

Ivan Čukić kdab.com, cukic.co

DISCLAIMER

The code in this presentation is slideware,
with chunks written by AI.
Do not use in production.

INTRO TO
RECURSION
OF HAPPINESS
IN THE ZEN GARDEN
OF CHAINS

Ivan Čukić kdab.com, cukic.co

INTRO TO RECURSION OF HAPPINESS IN THE ZEN GARDEN OF CHAINS

Meeting C++ 2025

Ivan Čukić kdab.com, cukic.co

INTRO TO
RECURSION
OF HAPPINESS
IN THE ZEN GARDEN
OF CHAINS

Functions shall not call themselves,
either directly or indirectly.

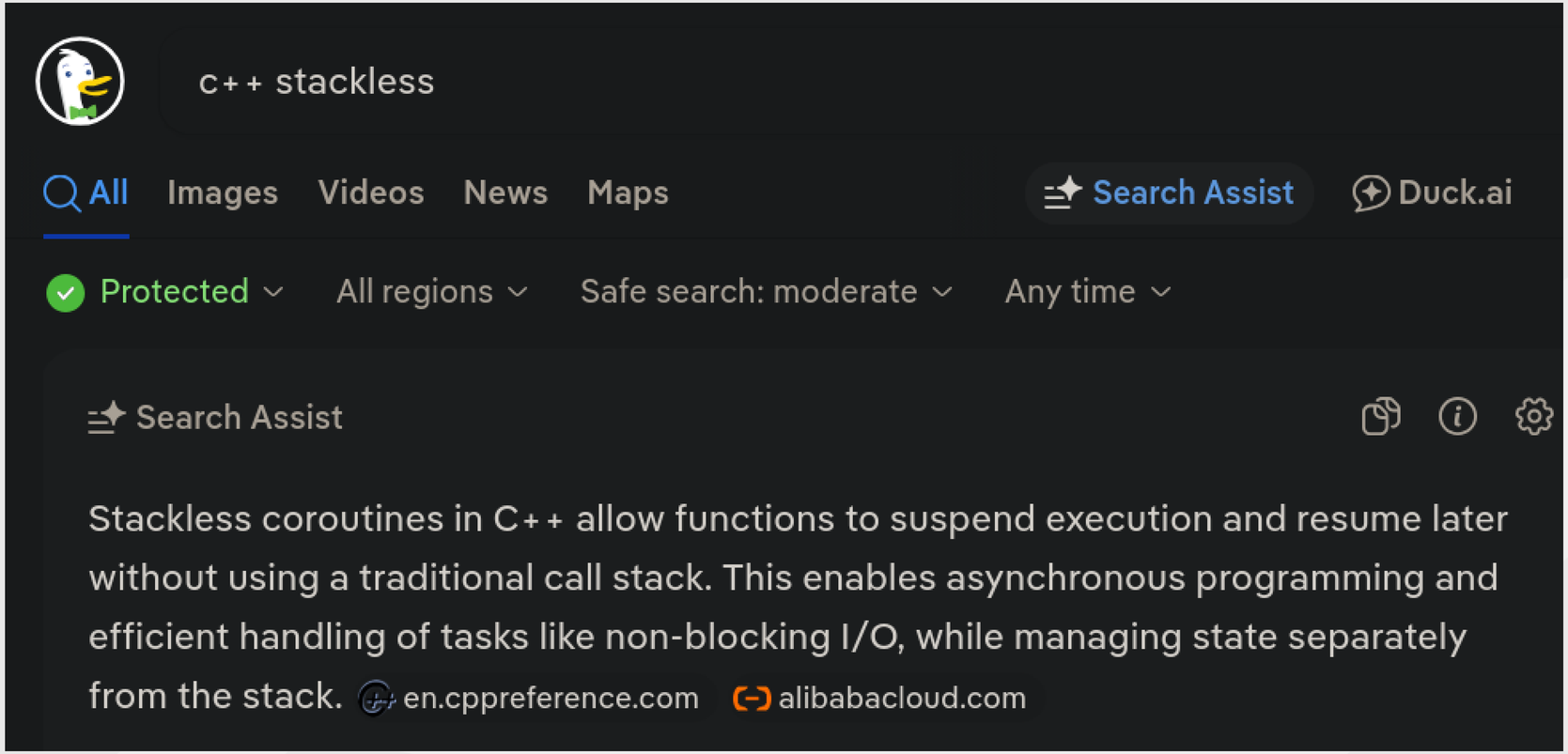
Variables local to a function are stored in the call stack.

If a function calls itself directly or indirectly several times,

the available stack space can be exceeded,

causing serious failure.

STACKLESS



INTRO TO
RECURSION
OF HAPPINESS
IN THE ZEN GARDEN
OF CHAINS

Meeting C++ 2025

Ivan Čukić kdab.com, cukic.co

RECURSION

```
co_recursion<int> fib(int n) {
    if (...) //
    auto last = co_await fib(n);
    auto penultimate = co_await fib(n - 1);
    co_return last + penultimate;
}
```

Done in collab with S. Malkov and P. Djordjević.

INTRO TO
RECURSION
OF HAPPINESS
IN THE ZEN GARDEN
OF CHAINS

Ivan Čukić kdab.com, cukic.co

NO SAINTS HERE

HALO – Heap Allocation eLision Optimization (P0981)

```
.
std::generator<int> range(int from, int to) {
    for (int i = from; i < to; ++i)
        co_yield i;
}

auto s = range(1, 10);
return std::accumulate(s.begin(), s.end(), 0);
```

INTRO TO
RECURSION
OF HAPPINESS
IN THE ZEN GARDEN
OF CHAINS

Ivan Čukić kdab.com, cukic.co

RECURSION

```
template<typename TResult>
struct co_recursion {
    struct promise_type {
        static void* operator new(std::size_t count) { ... }
        static void* operator new[](std::size_t count) { ... }
        static void operator delete(void* ptr, std::size_t sz) { ...
        static void operator delete[](void* ptr, std::size_t sz)  {
        ...
    };
};
```

INTRO TO
RECURSION
OF HAPPINESS
IN THE ZEN GARDEN
OF CHAINS

Ivan Čukić kdab.com, cukic.co

RECURSION

```
template<typename TResult>
struct co_recursion {
    struct promise_type {
        std::suspend_always initial_suspend()
            noexcept { return {}; }
        std::suspend_always final_suspend()
            noexcept { return {}; }
        ...
    };
};
```

INTRO TO
RECURSION
OF HAPPINESS
IN THE ZEN GARDEN
OF CHAINS

Ivan Čukić kdab.com, cukic.co

RECURSION

```
template<typename TResult>
struct co_recursion {
    bool await_ready() const {
        return false;
    }

    TResult await_resume() const;
    void await_suspend(coro_handle cont);
};
```

INTRO TO
RECURSION
OF HAPPINESS
IN THE ZEN GARDEN
OF CHAINS

Ivan Čukić kdab.com, cukic.co

RECURSION

```
template<typename TResult, typename THandle>
struct recursion_context {
    TResult result;
    std::stack<THandle> handles_to_resume;
};
```

INTRO TO
RECURSION
OF HAPPINESS
IN THE ZEN GARDEN
OF CHAINS

Ivan Čukić kdab.com, cukic.co

RECURSION

```
TResult await_resume() const {
    // result is set in promise_type::return_value
    return recursion_context->result;
}

void await_suspend(coro_handle cont) {
    recursion_context = cont.promise()
        .promise_recursion_context;
    recursion_context->handles_to_resume.push(cont);
}
```

INTRO TO
RECURSION
OF HAPPINESS
IN THE ZEN GARDEN
OF CHAINS

Ivan Čukić kdab.com, cukic.co

RECURSION

```
template<typename T>
T get(const co_recursion<T> &rec_def) {
    typename co_recursion<T>::context context;
    rec_def.context = std::addressof(context);
    context.handles_to_resume.push(rec_def.handle);

    while (!context.handles_to_resume.empty()) {
        ...
    }

    return context.result;
}
```

INTRO TO
RECURSION
OF HAPPINESS
IN THE ZEN GARDEN
OF CHAINS

Ivan Čukić kdab.com, cukic.co

RECURSION

```
while (!context.handles_to_resume.empty()) {
    auto &current_handle = context.handles_to_resume.top();
    if (current_handle.done()) {
        current_handle.destroy();
        context.handles_to_resume.pop();

    } else {
        current_handle.promise().promise_recursion_context =
            std::addressof(context);
        current_handle.resume();
    }
}
```

INTRO TO
RECURSION
OF HAPPINESS
IN THE ZEN GARDEN
OF CHAINS

Ivan Čukić kdab.com, cukic.co

DEFINITION

Definition (not) – Coroutines are a mechanism to redefine how (parts of) the C++-abstract machine works.

INTRO TO
RECURSION
OF HAPPINESS
IN THE ZEN GARDEN
OF CHAINS

Ivan Čukić kdab.com, cukic.co

HOOKS

```
.
struct logger {
    void increase_indent() { indent++; }
    void decrease_indent() { indent--; }
    void print(auto&& ... args) {
        log << std::string(indent * 4, ' ');
        // log the args
    }
};
```

INTRO TO
RECURSION
OF HAPPINESS
IN THE ZEN GARDEN
OF CHAINS

Ivan Čukić kdab.com, cukic.co

HOOKS

Custom contextual data for function execution.

Shared context for all, but also per-call context.

INTRO TO
RECURSION
OF HAPPINESS
IN THE ZEN GARDEN
OF CHAINS

Ivan Čukić kdab.com, cukic.co

DEFINITION

Definition (not) – Coroutines are a mechanism to hook into function call events and provide shadow contextual data.

INTRO TO RECURSION OF HAPPINESS IN THE ZEN GARDEN OF CHAINS

Meeting C++ 2025

Ivan Čukić kdab.com, cukic.co

INTRO TO RECURSION OF HAPPINESS IN THE ZEN GARDEN OF CHAINS

HAPPY PATH

C++ error handling, let’s abuse the co_await operator

Antoine Morrier

https://cpp-rendering.io/

INTRO TO
RECURSION
OF HAPPINESS
IN THE ZEN GARDEN
OF CHAINS

HAPPY PATH

```
co::expected<std::string, err_t> get_input();
co::expected<email_t, err_t> parse_email(const std::string&);

co::expected<std::string, err_t> generate_email_link() {
    std::string input = co_await get_input();
    auto email = co_await parse_email(input);
    auto html = format_link(email->user, email);
    co_return html;
}
```

INTRO TO
RECURSION
OF HAPPINESS
IN THE ZEN GARDEN
OF CHAINS

Ivan Čukić kdab.com, cukic.co

HAPPY PATH

```
template<typename T, typename E>
struct promise_expected {
    co::expected<T, E>* result_ptr = nullptr;
    co::expected<T, E> get_return_object();

    std::suspend_never initial_suspend() ...
    std::suspend_never final_suspend() ...

    void return_value(T value);
    void unhandled_exception() { ... }
};
```

INTRO TO
RECURSION
OF HAPPINESS
IN THE ZEN GARDEN
OF CHAINS

Ivan Čukić kdab.com, cukic.co

HAPPY PATH

```
template<typename T, typename E>
co::expected<T, E>
promise_expected<T, E>::get_return_object() {
    co::expected<T, E> result;
    result_ptr = &result; // !!!
    return result;
}
```

INTRO TO
RECURSION
OF HAPPINESS
IN THE ZEN GARDEN
OF CHAINS

Ivan Čukić kdab.com, cukic.co

HAPPY PATH

```
template<typename T, typename E>
void promise_expected<T, E>::return_value(T value) {
    if (result_ptr) {
        result_ptr->value_ = std::move(value);
    }
}
```

INTRO TO
RECURSION
OF HAPPINESS
IN THE ZEN GARDEN
OF CHAINS

Ivan Čukić kdab.com, cukic.co

HAPPY PATH

```
struct expected {
    struct awaiter {
        co::expected<T, E>& exp;

        bool await_ready() const noexcept {
            return exp.has_value();
        }

        T await_resume() {
            return *std::move(exp);
        }
```

INTRO TO
RECURSION
OF HAPPINESS
IN THE ZEN GARDEN
OF CHAINS

Ivan Čukić kdab.com, cukic.co

HAPPY PATH

```
struct expected {
    struct awaiter {
        void await_suspend(auto h) {
            if (h.promise().result_ptr) {
                h.promise().result_ptr->value_ =
                    std::unexpected(exp.error());
            }
            h.destroy();
        }
    };
```

INTRO TO
RECURSION
OF HAPPINESS
IN THE ZEN GARDEN
OF CHAINS

Ivan Čukić kdab.com, cukic.co

HAPPY PATH

```
co::expected<int, err_t> f() {
    auto* _obj =
        co_await co_dynamic_cast<SubClass*>(obj);
    ...
    return _obj->size();
}
```

INTRO TO
RECURSION
OF HAPPINESS
IN THE ZEN GARDEN
OF CHAINS

Ivan Čukić kdab.com, cukic.co

INTRO TO RECURSION OF HAPPINESS IN THE ZEN GARDEN OF CHAINS

Meeting C++ 2025

Ivan Čukić kdab.com, cukic.co

INTRO TO
RECURSION
OF HAPPINESS
IN THE ZEN GARDEN
OF CHAINS

MANY PATHS

```
struct inhale_t { int depth; quality_t quality; };
struct peak_t { seconds stillness; };
struct exhale_t { seconds release; };
struct rest_t { seconds stillness; string thought; };

using state = co::variant<
    inhale_t, peak_t, exhale_t, rest_t>;
```

INTRO TO
RECURSION
OF HAPPINESS
IN THE ZEN GARDEN
OF CHAINS

Ivan Čukić kdab.com, cukic.co

MANY PATHS

```
state_t inhale_update(state_t state)
{
    auto inhaling = co_await co_get<inhale_t>(state);

    inhaling.depth++;

    co_return inhaling;
}
```

INTRO TO
RECURSION
OF HAPPINESS
IN THE ZEN GARDEN
OF CHAINS

Ivan Čukić kdab.com, cukic.co

MANY PATHS

```
state_t inhale_to_peak(state_t state)
{
    auto _ = co_await co_get<inhale_t>(state);

    co_return peak_t{12};
}
```

INTRO TO
RECURSION
OF HAPPINESS
IN THE ZEN GARDEN
OF CHAINS

Ivan Čukić kdab.com, cukic.co

MANY PATHS

```
state_t peak_update(state_t state)
{
    auto inhaling = co_await co_get<peak_t>(state);

    inhaling.stillness++;

    co_return inhaling;
}
```

INTRO TO
RECURSION
OF HAPPINESS
IN THE ZEN GARDEN
OF CHAINS

Ivan Čukić kdab.com, cukic.co

MANY PATHS

```
auto breathing_cycle =
        fn::compose(
            peak_update,
            inhale_to_peak,
            inhale_update,
            ...)
```

INTRO TO
RECURSION
OF HAPPINESS
IN THE ZEN GARDEN
OF CHAINS

Ivan Čukić kdab.com, cukic.co

DEFINITION

Definition (not) – Coroutines are the most complex control flow structure in C++ that somehow turned out to be useful for asynchrony and lazy evaluation.

INTRO TO RECURSION OF HAPPINESS IN THE ZEN GARDEN OF CHAINS

Meeting C++ 2025

Ivan Čukić kdab.com, cukic.co

INTRO TO RECURSION OF HAPPINESS IN THE ZEN GARDEN OF CHAINS

PARSER COMBINATORS

https://github.com/no-more-secrets/parsco

.

```
parsco::parser<char> parse_two_same() {
    char c1 = co_await parsco::any_chr();
    char c2 = co_await parsco::any_chr();
    if (c1 != c2) {
        co_await parsco::fail( "expected two equal chars" );
    }
    co_return c1;
}
```

INTRO TO
RECURSION
OF HAPPINESS
IN THE ZEN GARDEN
OF CHAINS

Ivan Čukić kdab.com, cukic.co

QT TASKS

https://qcoro.dev by Daniel Vratil

.

```
QCoro::Task<> MyClass::fetchData() {
    QNetworkReply nam;
    auto *reply = co_await nam.get(...);
    const auto data = co_await reply->readAll();
    reply->deleteLater();
    doSomethingWithData(data);
}
```

INTRO TO
RECURSION
OF HAPPINESS
IN THE ZEN GARDEN
OF CHAINS

DEFINITION

Definition (not) – Coroutines are a way to chain weird kinds of functions in C++.

INTRO TO
RECURSION
OF HAPPINESS
IN THE ZEN GARDEN
OF CHAINS

Ivan Čukić kdab.com, cukic.co

COMPLEXITY

- Functions that can be suspended
- Functions on a custom abstract machine
- Way to register hooks for function invocation
- Complex control flow structure
- State machines
- Parser combinators
- Monads?

INTRO TO
RECURSION
OF HAPPINESS
IN THE ZEN GARDEN
OF CHAINS

Ivan Čukić kdab.com, cukic.co

BOOKS

Save Half on your entire purchase when you buy any 2 or more of these books!

https://www.manning.com/bundles/Meeting-C-2025

(Fran's, Anthony's, mine, ...)