goto; chicago





Click ‘Rate Session’

Rate **5** sessions to get the supercool GOTO reward

follow us @gotochgo

goto; chicago



THE NEWEST OLD LANGUAGE

MATT GODBOLT

goto; chicago

WHO USES C++?

- Games
- Finance
- Embedded devices
- Servers

BUT WHY C++?

goto; chicago

C++ STRENGTHS

- Value types
- Clear object lifetimes
- Performance

"But isn't C++ the language with the horrible error messages and all the crashing?"


goto;
chicago
That's old C++!
New C++ is better!
follow us @gotochgo

goto; chicago

So, what's changed?

Era	Highlights
C++98	memory leaks, use-after-free, crashes...
C++11	`auto`, range-`for`, lambdas, move, smart pointers, `constexpr`, atomics, UDLs...
C++14	return type deduction, better lambdas
C++17	`if constexpr`, `optional`, `variant`, `string_view`
C++20	Concepts, ranges, coroutines... maybe?

```
double rms(const vector<double> &v) {
  double sum = 0.0;
  for (vector<double>::const_iterator i = v.begin();
       i != v.end();
       i++) {
    sum += *i * *i;
  }
  return sqrt(sum);
}
```

```
double rms(const vector<double> &v) {
  double sum = 0.0;
  for (auto i = v.begin(); i != v.end(); i++) {
    sum += *i * *i;
  }
  return sqrt(sum);
}
```

```
double rms(const vector<double> &v) {
  double sum = 0.0;
  for (auto i : v)
    sum += i * i;
  return sqrt(sum);
}
```

```
static double accum(double partialSum, double elem) {
    return partialSum + elem * elem;
}
double rms(const vector<double> &v) {
  return sqrt(accumulate(
        v.begin(), v.end(), 0.0, accum));
}
```

```
double rms(const vector<double> &v) {
  return sqrt(accumulate(
      v.begin(), v.end(), 0.0,
      [](double partialSum, double elem) {
          return partialSum + elem * elem;
      }
  ));
}
```


goto;
chicago
WHY C++?
follow us @gotochgo
10

VALUE TYPES

- Passed by value
- Zero overhead
- Position, error code, prices, quantities, times...
- Strong typing

```
void buyShares(const char *symbol,
               double price,
               int quantity);

buyShares("GOOG", 800, 975.0); // whoops!
```

```
void buyShares(const char *symbol,
               Price price,
               Quantity qty);

buyShares("GOOG", Quantity(800), Price(975.0));
```

```
error: no matching function for call to 'buyShares'
    buyShares("GOOG", Quantity(800), Price(975.0));
    ^~~~~~~~~
note: candidate function not viable: no known conversion from
    'Quantity' to 'Price' for 2nd argument
```

goto; chicago

VALUE TYPES

```
class Pos {
  float x{};
  float y{};
public:
  constexpr Pos() = default;
  constexpr Pos(float x, float y)
    : x(x), y(y) {}
  constexpr Pos operator +(Pos other) const {
      return Pos(x + other.x, y + other.y);
  }
  // and so on for other operators...
};
```

CLEAR OBJECT LIFETIMES

```
string queryDb(const string &q) {
  DbConnection conn("server", 9000);
  return conn.query(q);
}
```

goto; chicago

CLEAR OBJECT LIFETIMES

```
class Document {
  vector<unique_ptr<Elem>> objects_;
public:
  void add(unique_ptr<Elem> &&object) {
    objects_.emplace_back(move(object));
  }
};
```

PERFORMANCE

```
auto midPoint(const vector<Pos> &vec) {
  auto total = accumulate(begin(vec), end(vec), Pos());
  return total / vec.size();
}
```

Compiler Explorer link


goto;
chicago
BUT WAIT, THERE'S MORE!
follow us @gotochgo
16

COMPILERS

- Clang
- GCC
- icc
- MSVC

goto; chicago

SANITIZERS

- Memory accesses
- Undefined behaviour
- Thread safety

goto; chicago

SANITIZERS

```
int main(int argc, const char *argv[]) {
   char array[10]{};
   return array[11];
}
```

SANITIZERS

goto; chicago

```
matthew@danger ~/d/goto-2018-cpp (master)> g++ -fsanitize=address test.cc
matthew@danger ~/d/goto-2018-cpp (master)> ./a.out
=================================================================
==2422==ERROR: AddressSanitizer: stack-buffer-overflow on address 0x7ffec9c26eab at pc 0x55bcd1079ab3 bp
0x7ffec9c26e60 sp 0x7ffec9c26e50
READ of size 1 at 0x7ffec9c26eab thread T0
    #0 0x55bcd1079ab2 in main (/home/matthew/dev/goto-2018-cpp/a.out+0xab2)
    #1 0x7f66b202a1c0 in __libc_start_main (/lib/x86_64-linux-gnu/libc.so.6+0x211c0)
    #2 0x55bcd1079899 in _start (/home/matthew/dev/goto-2018-cpp/a.out+0x899)

Address 0x7ffec9c26eab is located in stack of thread T0 at offset 43 in frame
    #0 0x55bcd1079989 in main (/home/matthew/dev/goto-2018-cpp/a.out+0x989)

  This frame has 1 object(s):
    [32, 42) 'array' <== Memory access at offset 43 overflows this variable
HINT: this may be a false positive if your program uses some custom stack unwind mechanism or swapcontext
      (longjmp and C++ exceptions *are* supported)
```

goto; chicago

SMART EDITORS

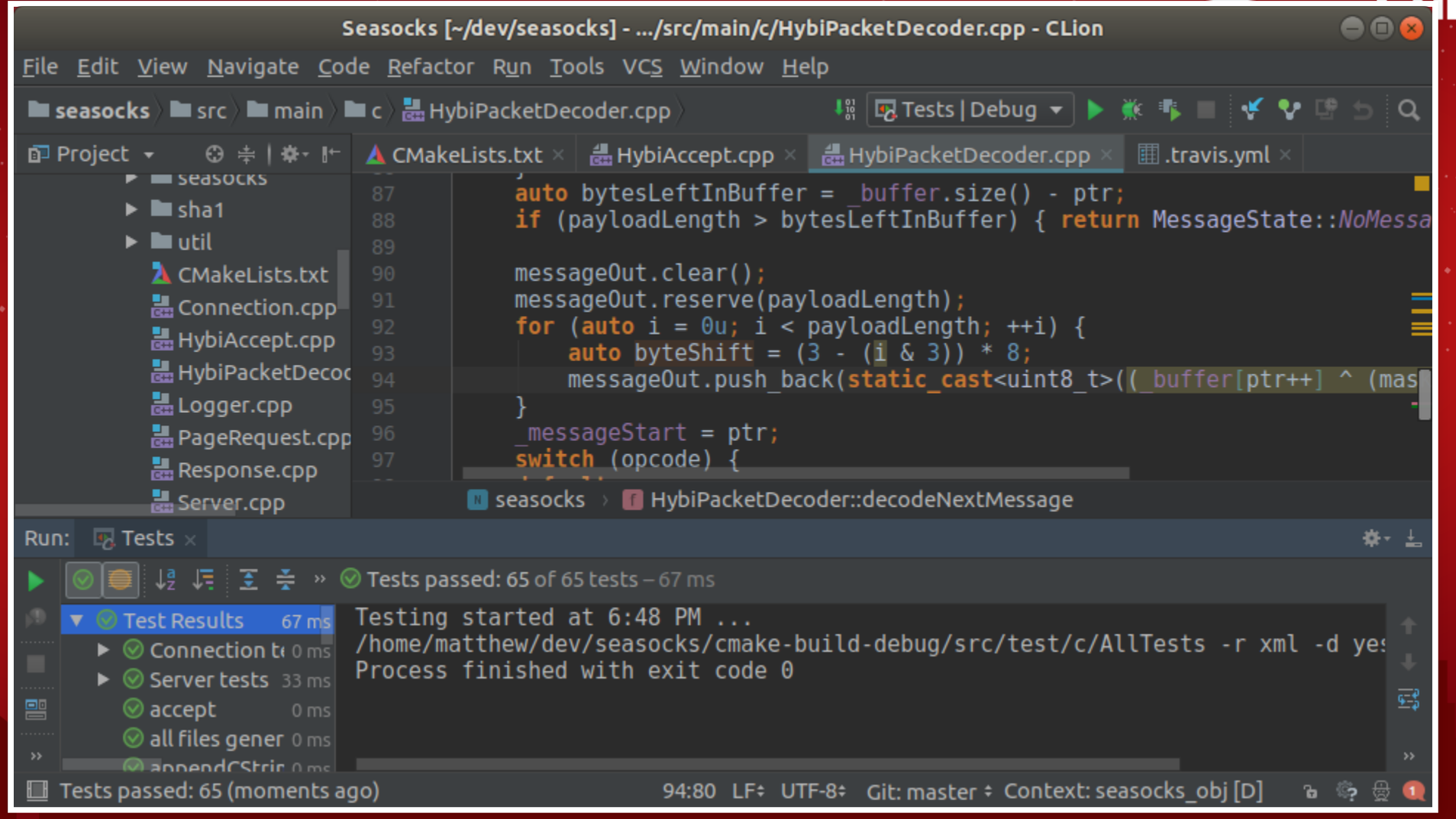
- vim/emacs & YouCompleteMe
- JetBrains CLion
- Eclipse CDT

SMART EDITORS

goto; chicago

```
    // From Response:
    virtual void handle(shared_ptr<ResponseWriter> writer) override {
        auto &server = _server;
        thread t([&server, writer] () mutable {
            usleep(1000000); // A long database query...
            string response = "some kind of response...beginning<br>";
            server.execute([response, writer]{
                writer->begin(ResponseCode::Ok, TransferEncoding::Chunked);
                writer->header("Content-type", "application/html");
                writer->payload(response.data(), response.length());
                writer->
            });
            response =  begin( ResponseCode responseCode, TransferEncoding encoding = TransferEncoding::R
            for (auto i error( ResponseCode responseCode, const std::string &payload )
                usleep( finish( bool keepConnectionOpen )
                server. header( const std::string &header, const std::string &value )
                    wri isActive() const
                }); operator=( const ResponseWriter & )
            } payload( const void *data, size_t size, bool flush = true )
            response = ResponseWriter::
            usleep(100000); // final data ~ResponseWriter()
            server.execute([response, writer]{
                writer->payload(response.data(), response.length());
                writer- begin( ResponseCode responseCode, TransferEncoding encoding = TransferEncoding::R
-- INSERT --
```

SMART EDITORS



COMMUNITY SUPPORT

- Core guidelines
- Guideline support library
- #include<C++>
- Slack
- cppbestpractices

goto; chicago

WHAT ARE YOU WAITING FOR?

GIVE C++ A TRY NOW!



goto; conference



Please

Remember to rate this session

Thank you!