CINQ

C++ Integrated Query

A data processing library for C++

It’s pronounced “sink”

LINQ

Allows the programmer to “query” container objects

```
List<Person> contacts = ...;
var result = contacts.Where(p => p.Age >= 21)
                     .OrderBy(p => p.Age)
                     .ThenBy(p => p.LastName)
                     .ToList();
```

CINQ

We wanted to implement that, but for C++

```
vector<person> contacts = ...;
auto result = cinq::from(contacts)
    .where([](auto& p) { return p.age >= 21});
    .order_by([](auto& p) { return p.age },
              [](auto& p) { return p.last_name; })
    .to_vector();
```

Why is it useful?

```
vector<weather_point> result;
for (auto& data : weather_data)
{
     if (data.rain) result.push_back(data);
}

sort(result.begin(), result.end(),
    ](const auto &a, const auto &b) { return a.temp_min < b.temp_min; });

vector<weather_point> five;
for (size_t i = 0; i < 5; i++) five.push_back(result[i]);

vector<int> temps;
for (auto& data : five) temps.push_back(data.temp_min);
```

Why is it useful?

```
cinq::from(weather_data)
     .where([](const weather_point& w) { return w.rain; })
     .order_by([](const weather_point& w) { return w.temp_min; })
     .take(5)
     .select([](const weather_point& w) { return w.temp_min; })
     .to_vector();
```

Existing implementations?

There are already several attempts at writing a LINQ library for C++, for instance:

- boost.range http://www.boost.org/doc/libs/1_54_0/libs/range/
- Linq http://pfultz2.github.io/Linq/
- boolinq http://code.google.com/p/boolinq/
- lx++ (part of Native-RX) https://rx.codeplex.com/
- Linq++ https://github.com/hjiang/linqxx/
- oven http://p-stade.sourceforge.net/oven/
- cpplinq (oops same name but completely unrelated http://code.google.com/p/cpplinq/
- linq to object http://www.cnblogs.com/cbscan/archive/2012/10/20/2732773.html
- linq https://bitbucket.org/ronag/cppextras/src/master/linq/linq.hpp
- Streams http://www.infoq.com/news/2014/07/cpp14-streams-lazy-functional

Source: https://cpplinq.codeplex.com

It's OK, cpplinq is not as pretty

- A lot of typing
- Unconstrained templates produce messy error messages
- We didn't want to abuse the bit shift operators any more

:(

```
auto result = from_array(contacts)
>> where([](person p) {return p.age >= 21;})
>> orderby_ascending([](auto p){return p.age;});
>> thenby_ascending([](auto p){return p.last_name;})
>> to_vector();
```

:(

:(

CINQ Goals

- Improved syntax
- Constrain templates with concepts
- Learn about templates & concepts

Features

Methods implemented

- Select
- Where
- Single
- Any, All
- Min, Max, Sum, Average
- Take
- Skip
- ElementAt, First, Last
- Concat
- OrderBy
- Reverse

Better compiler error messages

```
cinq_enumerable.hpp: In instantiation of ‘cinq::enumerable<TSource> cinq::enumerable<TSource,
TElement, TIter>::where(TFunc) [with TFunc = make_tests():<lambda()>:<lambda(int)>; TSource
= std::vector<int>; TElement = int; TIter = __gnu_cxx::__normal_iterator<const int*,
std::vector<int> >; typename TSource::const_iterator = __gnu_cxx::__normal_iterator<const
int*, std::vector<int> >; typename TSource::value_type = int]’:
cinq_test.cpp:59:57:   required from here
cinq_enumerable.hpp:58:38: error: no matching function for call to
‘cinq::enumerable<std::vector<int>, int, __gnu_cxx::__normal_iterator<const int*,
std::vector<int> > >::where(make_tests():<lambda()>:<lambda(int)>&,
std::vector<int>::const_iterator, std::vector<int>::const_iterator)’
            if (is_data_copied) where(predicate, data.cbegin(), data.cend());
                                ^

cinq_enumerable.hpp:81:14: note: candidate: void cinq::enumerable<TSource, TElement,
TIter>::where(TFunc, TIterator, TIterator) [with TFunc =
make_tests():<lambda()>:<lambda(int)>; TIterator = __gnu_cxx::__normal_iterator<const int*,
std::vector<int> >; TSource = std::vector<int>; TElement = int; TIter =
__gnu_cxx::__normal_iterator<const int*, std::vector<int> >]
        void where(TFunc predicate, TIterator begin, TIterator end)
             ^

cinq_enumerable.hpp:81:14: note:   constraints not satisfied
cinq_enumerable.hpp:81:14: note:   concept ‘Predicate<TFunc, int, size_t>()’ was not satisfied
```

Better compiler error messages

- CINQ tells you *exactly* what you did wrong
- Spend less time fiddling with templates

```
constraints not satisfied
concept ‘Predicate<TFunc, int, size_t>()’ was not satisfied
```

So many tests

```
build@build-gcc:~/cinq/src$ ./cinq_test
[     ] where() std::vector
[     ] where() std::list
[     ] where() std::array
[     ] where() with index std::vector
[     ] where() with index std::list
[     ] where() with index std::array
[     ] any() true unit test
[     ] any() false unit test
[     ] any(Predicate) true unit test
[     ] any(Predicate) false unit test
[     ] concat() std::vector
[     ] concat().where() std::vector<string>
[     ] reverse() std::vector
[     ] select() std::vector
[     ] select() with index std::vector<int>
[     ] select() with index std::vector<string>
[     ] count() std::array
[     ] count() std::vector
[     ] count() std::list
[     ] all() std::vector
[     ] all() std::array
[     ] all() std::list
[     ] count(void) std::array
[     ] count(void) std::vector
[     ] count(void) std::list
[     ] take(int) std::vector
[     ] take(int) std::vector ensure_data
[     ] take(int) std::vector ensure_data, count > size
[     ] take(int) std::vector count > size
[     ] take(int) std::list
[     ] take(int) std::array
[     ] take(string).where() std::list
[     ] take(string /*literals*/).where() std::vector ;
[     ] first() std::vector;
[     ] first() ensure_data std::vector;
[     ] first(predicate) ensure_data std::vector str;
[     ] first(predicate) std::vector str;
[     ] first(predicate) std::list ints;
[     ] first(predicate) ensure_data std::list doubles;
[     ] last(predicate) ensure_data std::vector strings;
[     ] last()  std::list int;
[     ] last() ensure_data std::list int;
[     ] single() ensure_data std::list string
[     ] single()  std::list string
[     ] single()  std::vector string
[     ] single(predicate) ensure_data std::vector string
[     ] single(predicate) std::vector string
[     ] skip(string) std::vector string
[     ] skip(string) ensure_data std::vector string
[     ] skip(string) std::list string
[     ] orderby(2 lambdas), std::vector int
[     ] orderby(2 lambdas),  std::vector string
[     ] orderby(void),  std::vector int
[     ] orderby(void),  std::vector string
[     ] max() on int
[     ] max() on double
[     ] max() on strings with mapping function
[     ] min() on int
[     ] min() on double
[     ] min() on strings with mapping function
[     ] sum() on int
[     ] sum() on double
[     ] sum() on strings with mapping function
[     ] average() on int
[     ] average() on float
[     ] average() on string length with mapping function
[     ] average() on float with mapping function
67 tests passed :-)
[ 145] 2000x where() by temperature
[ 145] 2000x where() by temperature - manual
[ 191] 2000x select() mapping weather_point to cloud_cover
[ 162] 2000x select() mapping weather_point to cloud_cover - manual
[ 223] 500x where().average() finding the averge cloud_cover between 1980 and 2000
[ 129] 500x where().average() finding the averge cloud_cover between 1980 and 2000 - manual
[ 203] 130000000x max(). finding the max temp_max in the data set
[ 211] 130000000x max(). finding the max temp_max in the data set - manual
[ 198] 130000000x min(). finding the min temp_min in the data set
[ 295] 2000x where().select(). get a vector of temp_mins for the days that it snowed
[ 217] 2000x where().select(). get a vector of temp_mins for the days that it snowed - manual
[ 265] 100x where().select().order_by().take() - 5 coldest rainy days
[ 195] 100x where().select().order_by().take() - 5 coldest rainy days - manual
build@build-gcc:~/cinq/src$
```


```
[     ] min() on strings with mapping function
[     ] sum() on int
[     ] sum() on double
[     ] sum() on strings with mapping function
[     ] average() on int
[     ] average() on float
[     ] average() on string length with mapping function
[     ] average() on float with mapping function
67 tests passed :-)
```


```
[ 145] 2000x where() by temperature
[ 145] 2000x where() by temperature - manual
[ 191] 2000x select() mapping weather_point to cloud_cover
[ 162] 2000x select() mapping weather_point to cloud_cover - manual
[ 223] 500x where().average() finding the averge cloud_cover between 1980 a
[ 129] 500x where().average() finding the averge cloud_cover between 1980 a
[ 203] 130000000x max(). finding the max temp_max in the data set
[ 211] 130000000x max(). finding the max temp_max in the data set - manual
[ 198] 130000000x min(). finding the min temp_min in the data set
[ 295] 2000x where().select(). get a vector of temp_mins for the days that
[ 217] 2000x where().select(). get a vector of temp_mins for the days that
[ 265] 100x where().select().order_by().take() - 5 coldest rainy days
[ 195] 100x where().select().order_by().take() - 5 coldest rainy days - man
```

Interface Design

Combining order_by and then_by

```
myList.OrderBy(Person p => p.Age)
      .ThenBy(p => p.LastName)
      .ThenBy(p => p.FirstName);
```

Combining order_by and then_by

```
myList.OrderBy(Person p => p.Age)
      .ThenBy(p => p.LastName)
      .Where(p => p.Age >= 30)
      .ThenBy(p => p.FirstName);
```

Combining order_by and then_by

```
error CS0411: The type arguments for method
System.Linq.Enumerable.ThenBy<TSource,TKey>(this
System.Linq.IOrderedEnumerable<TSource>,
System.Func<TSource,TKey>) cannot be inferred from
the usage. Try specifying the type arguments
explicitly
```

Combining order_by and then_by

- Combining these two methods makes it impossible to get confused
- Better optimization opportunities since we have all the lambdas at once

```
cinq::from(my_vector)
     .order_by([](auto p) { return p.age; },
               [](auto p) { return p.last_name; },
               [](auto p) { return p.last_name; });
```

Combining order_by and then_by

```
template<typename ... TFunc>
enumerable<TSource> order_by(TFunc... rest)
{
    ensure_data();
    std::stable_sort(data.begin(), data.end(), multicmp(rest...));
    return *this;
}
```

Combining order_by and then_by

```
template<typename ... TFunc,
         typename TFirst = typename std::tuple_element<0, std::tuple<TFunc...>>::type,
         typename TReturn = typename result_of<TFirst(TElement)>::type>
requires Invokable<TFirst, TElement>() && Totally_ordered<TReturn>()
auto multicmp(TFirst first, TFunc... rest)
{
    return [=](const TElement& a, const TElement& b) -> bool
    {
        auto a_map = first(a);
        auto b_map = first(b);
        if (a_map == b_map) return multicmp(rest...)(a, b);
        else return a_map < b_map;
    };
}
```

Combining order_by and then_by

```
template<typename TFirst, typename TReturn = typename result_of<TFirst(TElement)>::type>
requires Invokable<TFirst, TElement>() && Totally_ordered<TReturn>()
auto multicmp(TFirst first)
{
    return [=](const TElement& a, const TElement& b) -> bool
    {
        return first(a) < first(b);
    };
}
```

size_t conversion errors

- size_t is the convention for C++ indexing
- size_t is a typedef for an unsigned integer
- No warnings when you do this:

```
cinq::from(my_vector).take( -1 );
```

size_t conversion errors

- size_t is the convention for C++ indexing
- size_t is a typedef for an unsigned integer
- No warnings when you do this:

```
int temp;
cin >> temp;
cinq::from(my_vector).take(temp);
```

size_t conversion errors

Problem, meet solution:

```
enumerable<TSource> take(int count)
{
    if (count >= 0) return take((size_t)count);
    else throw invalid_argument("cinq: take() was
called with negative count");
}
```

Default versions of methods

You can use a mapping lambda…

```
cinq::from(class_grades).average([](float grade)
{ return grade; });
```

…but we provide a default:

```
cinq::from(class_grades).average();
```

Concepts ensure you can only call if class_grades contains numbers

Default versions of methods

```
public static double Average (this IEnumerable<int> source)
{
    Check.Source (source);

    long total = 0;
    int count = 0;
    foreach (var element in source){
        total = checked (total + element);
        count++;
    }
    if (count == 0)
        throw EmptySequence ();
    return total / (double) count;
}
```

Source: mono project version effa4c0, file mcs/class/System.Core/System.Linq/Enumerable.cs

Default versions of methods

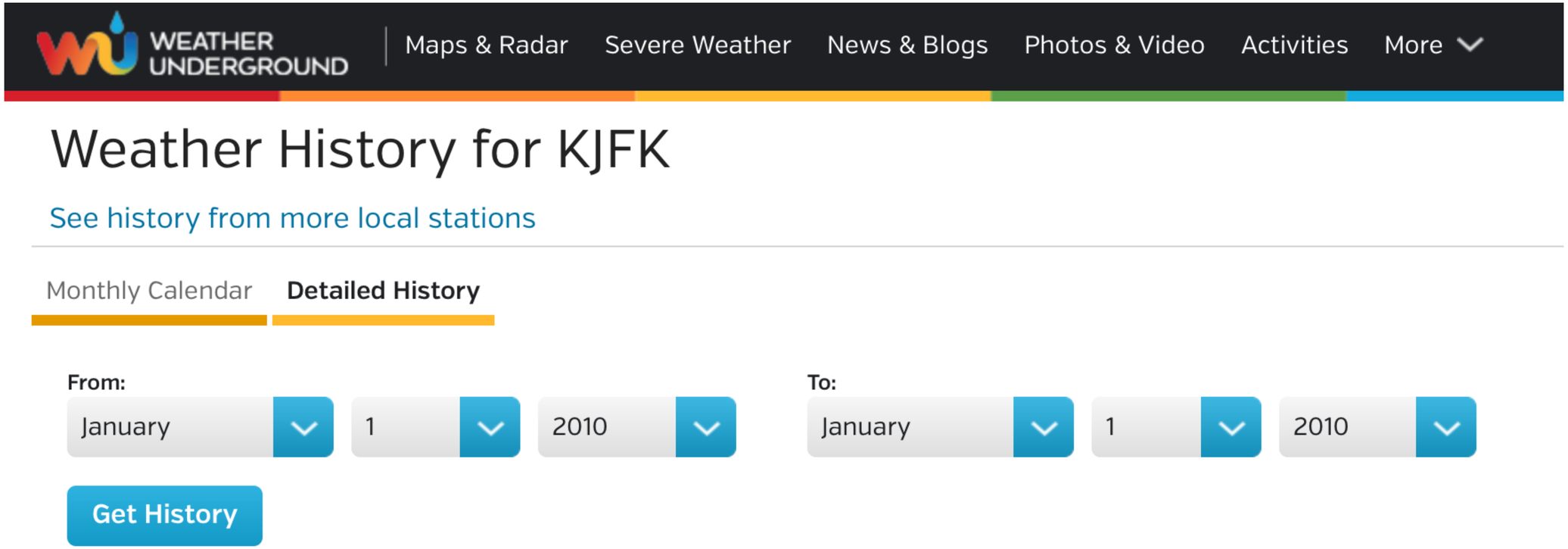
```
public static double   Average (this IEnumerable<int> source) {...}
public static double   Average (this IEnumerable<long> source) {...}
public static double   Average (this IEnumerable<double> source) {...}
public static float    Average (this IEnumerable<float> source) {...}
public static decimal  Average (this IEnumerable<decimal> source) {...}
public static double?  Average (this IEnumerable<int?> source) {...}
public static double?  Average (this IEnumerable<long?> source) {...}
public static double?  Average (this IEnumerable<double?> source) {...}
public static decimal? Average (this IEnumerable<decimal?> source) {...}
public static float?   Average (this IEnumerable<float?> source) {...}
```

Source: mono project version effa4c0, file mcs/class/System.Core/System.Linq/Enumerable.cs

Performance

Finding a large data set

- Weather Underground lets you export weather history year by year
- No protection against scraping!



Source: http://www.wunderground.com/history/airport/KJFK/2010/1/1/CustomHistory.html

Finding a large data set

- Wrote a program to download JFK airport data, 1950–present
- 23,982 rows (about 1 per day)

```
EST,Max TemperatureF,Mean TemperatureF,Min TemperatureF,Max Dew PointF,MeanDew PointF,Min
DewpointF,Max Humidity,Mean Humidity,Min Humidity,Max Sea Level PressureIn,Mean Sea Level
PressureIn,Min Sea Level PressureIn,Max VisibilityMiles,Mean VisibilityMiles,Min VisibilityMiles,Max
Wind SpeedMPH,Mean Wind SpeedMPH,Max Gust SpeedMPH,PrecipitationIn,CloudCover,Events,WindDirDegrees
1948-7-1,84,78,72,71,65,58,93,65,46,30.07,30.01,29.98,10,7,2,16,8,,0.00,0,Fog,264
1948-7-2,82,72,63,62,53,49,76,51,33,30.20,30.15,30.08,15,14,10,16,10,,0.00,0,,315
1948-7-3,78,71,64,66,58,53,84,62,42,30.17,30.12,30.04,15,10,5,14,6,,0.00,0,,203
1948-7-4,84,76,68,68,63,56,90,67,38,30.13,30.10,30.07,15,7,2,12,5,,0.00,0,Fog,198
1948-7-5,93,82,70,74,69,65,93,71,40,30.12,30.03,29.89,10,6,3,18,8,,0.00,0,Fog-Rain-Thunderstorm,218
1948-7-6,91,82,72,71,68,64,91,75,50,29.88,29.69,29.54,10,6,2,28,10,,0.00,0,Rain-Thunderstorm,244
1948-7-7,73,66,60,65,56,50,93,72,44,30.08,29.88,29.60,15,12,3,22,12,,0.00,0,Rain,29
1948-7-8,84,72,61,62,56,44,93,63,25,30.26,30.18,30.10,15,10,2,14,8,,0.00,0,Fog,254
1948-7-9,81,72,64,64,59,57,90,64,44,30.30,30.27,30.24,10,8,2,18,10,,0.00,0,Fog,206
1948-7-10,82,73,64,65,61,56,84,67,39,30.26,30.20,30.12,15,9,3,18,11,,0.00,0,,208
1948-7-11,82,75,68,69,66,62,87,75,58,30.12,30.05,29.97,10,8,4,16,12,,0.00,0,,203
1948-7-12,82,76,69,70,68,66,90,78,63,30.03,30.00,29.93,10,6,3,20,12,,0.00,0,Fog,198
1948-7-13,79,74,70,71,69,66,100,89,74,29.99,29.90,29.82,10,7,1,23,7,,0.00,0,Fog-Rain-Thunderstorm,176
1948-7-14,84,77,70,72,68,62,100,78,51,29.99,29.87,29.78,15,8,1,17,8,,0.00,0,Fog-Rain,5
```

Battle of the LINQs

Test machine: Intel i7-4850HQ

The contenders:

- CINQ on Linux 3.16.0-34-generic
- C++ “by hand” on Linux 3.16.0-34-generic
- Microsoft C# VM on Windows 8.1

where(): Find days hotter than 90°F

CINQ	142 ms
C++ “By Hand”	131 ms
C# LINQ	603 ms

Based on CINQ version f7e6cb1

select(): Mapping to cloud cover

CINQ	178 ms
C++ “By Hand”	160 ms
C# LINQ	1085 ms

Based on CINQ version f7e6cb1

where().average(): Average daily cloud cover

CINQ	207 ms
C++ “By Hand”	129 ms
C# LINQ	935 ms

Based on CINQ version f7e6cb1

max(): finding max temperature

CINQ	212 ms
C++ “By Hand”	209 ms
C# LINQ	393 ms

Based on CINQ version f7e6cb1

where().select(): coldest snowy days

CINQ	301 ms
C++ “By Hand”	248 ms
C# LINQ	530 ms

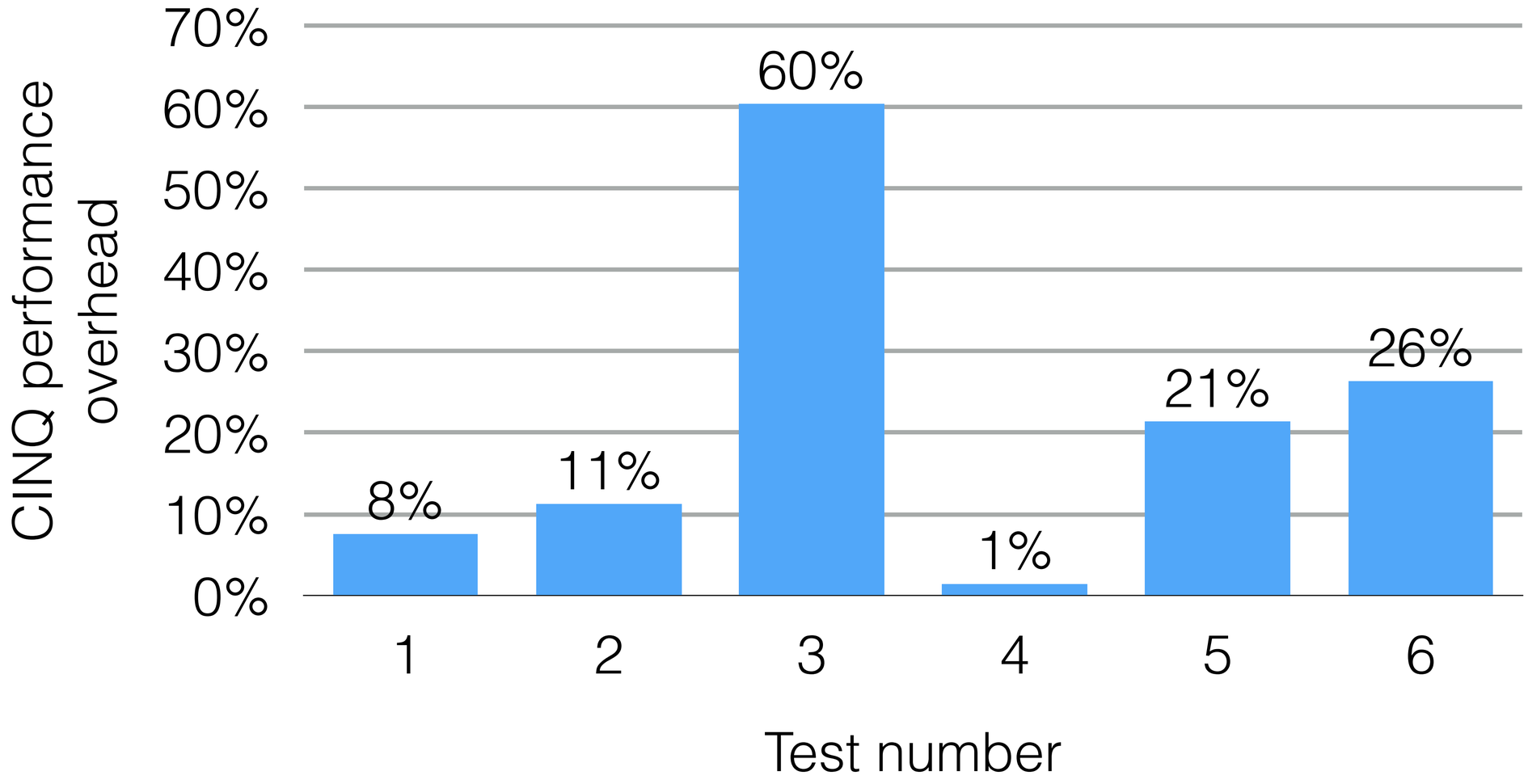
Based on CINQ version f7e6cb1

where().select().order_by().take(): 5 coldest rainy days

CINQ	254 ms
C++ “By Hand”	201 ms
C# LINQ	249 ms

Based on CINQ version f7e6cb1

Summary: CINQ vs “by hand”



Future Releases

Parallel queries

Automatically parallelize queries based on the number of CPUs

```
cinq::from(my_vector)
      .as_parallel()
      .where([](person p)
          { return some_very_expensive_call(p); });
```

Fully implement method syntax

- select_many
- join
- group_join
- distinct
- except
- intersect
- union
- group_by
- aggregate

Query syntax

Not too useful, but it looks cool

```
auto result = ( <from> my_contacts
                <where> [](person &p) { return p.age >= 21 }
                <order_by> ascending ).to_vector();
```

Questions?

Thanks for learning about our project.

Get the code at github.com/jeb2239/CINQ

Kevin Chen, Jonathan Barrios, Jonathan Wong