C++20 Ranges : Code expressively with Ranges

Harishankar Singh

QICKSTEP™
SET YOUR CAREER IN MOTION

Disclaimer

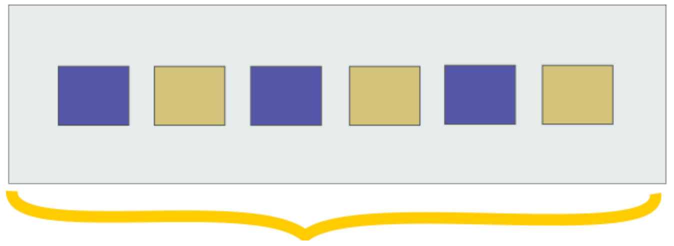
What all in C++20 Ranges?

- Ranges
- Range Algorithms
- Views

```
#include<ranges> //Header for Ranges and views

namespace std{
    namespace ranges{...}        //Range Algorithm and Views
    namespace ranges::views{...} //Range Adaptors
    namespace views = ranges::views; //Shortcut to adaptors
}
```

Ranges



- A range is a group of items that you can iterate over. It provides a begin iterator and an end sentinel.
- The containers of the STL are ranges(array, vectors, List, Map, etc).
- String Views, Span are Ranges.

Ranges Algorithm: Directly On Containers

- C++20 ranges supports STL algorithms
  - ✓ Can operate directly on containers; you don’t need iterators to specify a range

Until C++20

```
#include<vector>
#include<algorithm>
int main() {
  std::vector<int> myVec{2, 1, 3, 5, 4, 6, 7, 8, 9};
  std::sort(myVec.begin(),myVec.end());
}
```

Since C++20

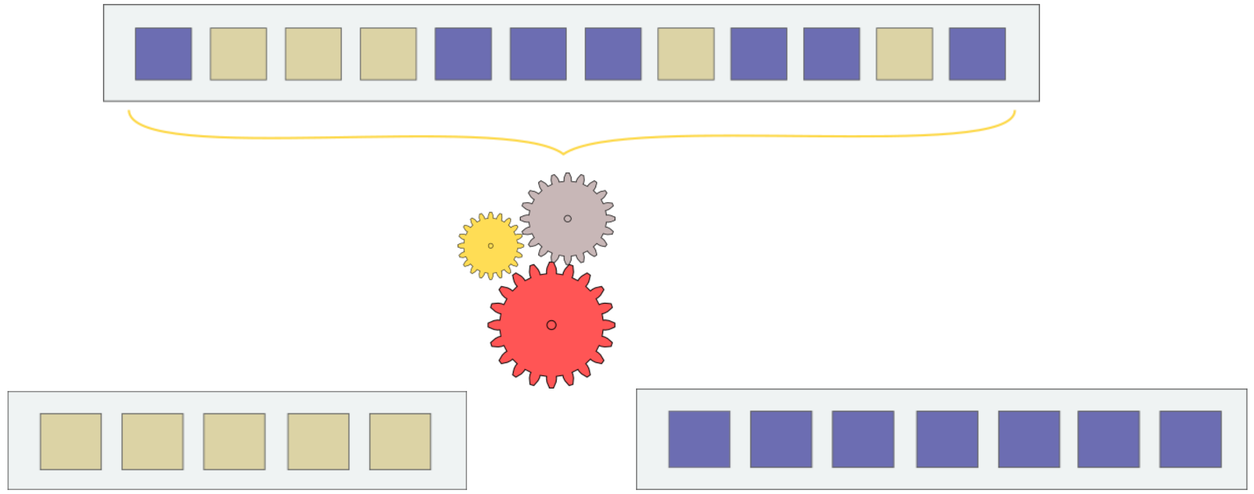
```
#include<vector>
#include<algorithm>
int main() {
  std::vector<int> myVec{2, 1, 3, 5, 4, 6, 7, 8, 9};
  std::ranges::sort(myVec);
}
```

Ranges: Functional Patterns

```cpp
#include <iostream>
#include <ranges>
#include <vector>
int main() {
  std::vector<int> myVec{1,2,3,4,5,6,7,8,9,10};
  for (int i : myVec | std::views::drop(5) | std::views::reverse) {
    std::cout << i << ' ';
  }
}
```

- Pipe symbol is for function composition. Read from left to right.
- "From Vector, drop first 5 elements and reverse remaining elements". A normal English sentence ☺

Views



- Views are ranges with Lazy evaluation.
- A view is something that you apply on a range and performs some operation.
- A view does not own data, and its time complexity to copy, move, or assign is constant(Cheap).

Views: Lazy Evaluation

```
#include <iostream>
#include <ranges>
#include <vector>
int main() {
  std::vector<int> myVec{1,2,3,4,5,6,7,8,9,10};
  auto myView = myVec | std::views::drop(5);
  std::cout<<*myView.begin()<<'\n';
}
```

- A key feature of views is that whatever transformation they apply, they do so at the moment you request an element, not when the view is created.
- myView is a view; creating it neither changes myVec, nor does myView store any elements. The time it takes to construct view and its size in memory is independent of the size of Vector.

Prime Numbers: The Sieve of Eratosthenes

1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100

- ✓ 1 is not a prime number, since its only factor is 1
- ✓ The first prime number and the only even prime number is 2. Since all other even numbers are divisible by 2, they cannot be primes, so all other prime numbers must be odd. So take your pencil and mark out all multiples of 2: 4, 6,.., 98, 100. That takes out 49 of the numbers!
- ✓ The next prime, 3, has only 2 factors, so all the other factors of 3 cannot be primes. So use your pencil to cross out 6 (it's already gone), 9,12 (already gone), 15, 21, 27, 33, 39, 45, 51, 57, 63, 69, 75, 81, 87, 93, 99.
- ✓ We continue this process, crossing out multiples of 5 (I found 25, 35, 65, 85, 95), multiples of 7 (49, 77, 91), 11 (none), 13 (none).

Prime Numbers: The Sieve of Eratosthenes

1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100

```cpp
#include <iostream>
#include <ranges>
#include <vector>
#include <numeric>
int main() {

   std::vector<int> numbers(100);
   std::iota(numbers.begin(),numbers.end(),1);
}
```

Prime Numbers: The Sieve of Eratosthenes

1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100

- ✓ It is clear that 1 is not a prime number, since its only factor is 1

```
#include <iostream>
#include <ranges>
#include <vector>
#include <numeric>
int main() {
 std::vector<int> numbers(100);
 std::iota(numbers.begin(),numbers.end(),1);
 auto results = numbers |std::views::filter([](int n){ return n!=1; });
}
```

Prime Numbers: The Sieve of Eratosthenes

	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100

```cpp
#include <iostream>
#include <ranges>
#include <vector>
#include <numeric>
int main() {
 std::vector<int> numbers(100);
 std::iota(numbers.begin(),numbers.end(),1);
 auto results = numbers
          |std::views::filter([](int n){ return n!=1; })
          |std::views::filter([](int n){ return n==2 || n % 2 != 0; });
}
```

✓ The first prime number and the only even prime number is 2. Since all other even numbers are divisible by 2, they cannot be primes, so all other prime numbers must be odd.

Prime Numbers: The Sieve of Eratosthenes

	2	3		5		7		9	
11		13		15		17		19	
21		23		25		27		29	
31		33		35		37		39	
41		43		45		47		49	
51		53		55		57		59	
61		63		65		67		69	
71		73		75		77		79	
81		83		85		87		89	
91		93		95		97		99	

```cpp
#include <iostream>
#include <ranges>
#include <vector>
#include <numeric>
int main() {
 std::vector<int> numbers(100);
 std::iota(numbers.begin(),numbers.end(),1);
 auto results = numbers
         |std::views::filter([](int n){ return n!=1; })
         |std::views::filter([](int n){ return n==2 || n % 2 != 0; })
         |std::views::filter([](int n){ return n==3 || n % 3 != 0; });

}
```

- ✓ The next prime, 3, has only 2 factors, so all the other factors of 3 cannot be primes. So use your pencil to cross out 6 (it’s already gone), 9,12 (already gone), 15, 21, 27, 33, 39, 45, 51, 57, 63, 69, 75, 81, 87, 93, 99.

Prime Numbers: The Sieve of Eratosthenes

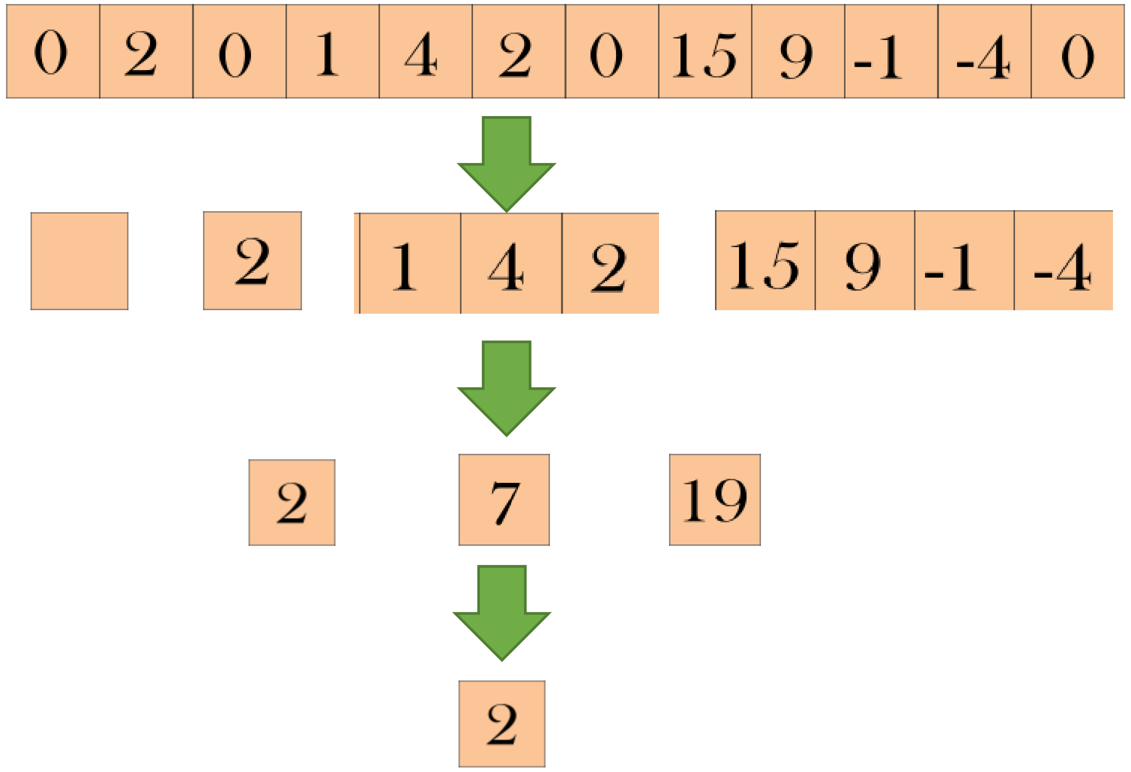
	2	3		5		7			
11		13				17		19	
		23						29	
31						37			
41		43				47		49	
		53						59	
61						67			
71		73				77		79	
		83						89	
91						97			

```cpp
#include <ranges>
#include <vector>
#include <numeric>
int main() {
 std::vector<int> numbers(100);
 std::iota(numbers.begin(),numbers.end(),1);
 auto results = numbers
         |std::views::filter([](int n){ return n!=1; })
         |std::views::filter([](int n){ return n==2 || n % 2 != 0; })
         |std::views::filter([](int n){ return n==3 || n % 3 != 0; });
         |std::views::filter([](int n){ return n==5 || n % 5 != 0; })
         |std::views::filter([](int n){ return n==7 || n % 7 != 0; });
}
```

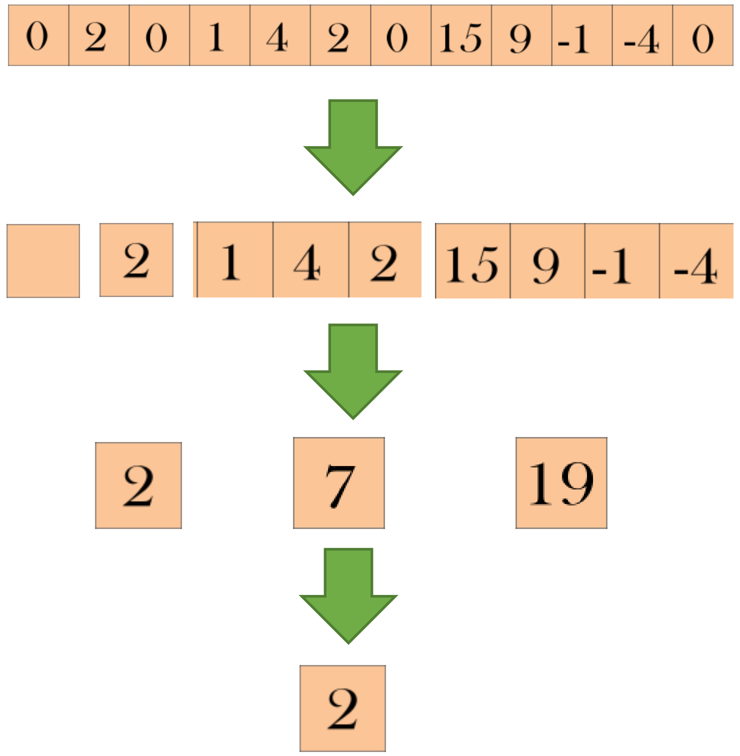
✓ We continue this process, crossing out multiples of 5 (I found 25, 35, 65, 85, 95), multiples of 7 (49, 77, 91), 11 (none), 13 (none).

Sum: The Even Sum of Numbers

✓ Find all the sub sequences between the pair zeros where the sum of those sub sequences is an even number.



Sum: The Even Sum of Numbers



```cpp
#include <algorithm>
#include <iostream>
#include <ranges>
#include <numeric>

int main()
{
   std::array source = {0,2,0,1,4,2,0,15,9,-1,-4,0 };
   int delimiter {0};

   std::ranges::lazy_split_view view {source, delimiter};

   auto evenSum = view|std::views::filter(Size)|std::views::filter(evenSumFilter);

   for (auto const& evenView: evenSum)
       print(evenView);
}
```

Sum: The Even Sum of Numbers

```cpp
auto evenSumFilter = [](auto const& view)
{
   return (sum(view)%2 == 0)?true:false;
};
```

```cpp
auto sum = [](auto const& view)
{
   auto a= view|std::views::common;
   return std::accumulate(a.begin(),a.end(),0);
};
```

```cpp
auto Size = [](auto const& view)
{
   auto a= view|std::views::common;
   return !(std::ranges::empty(a))?true:false;
};
```

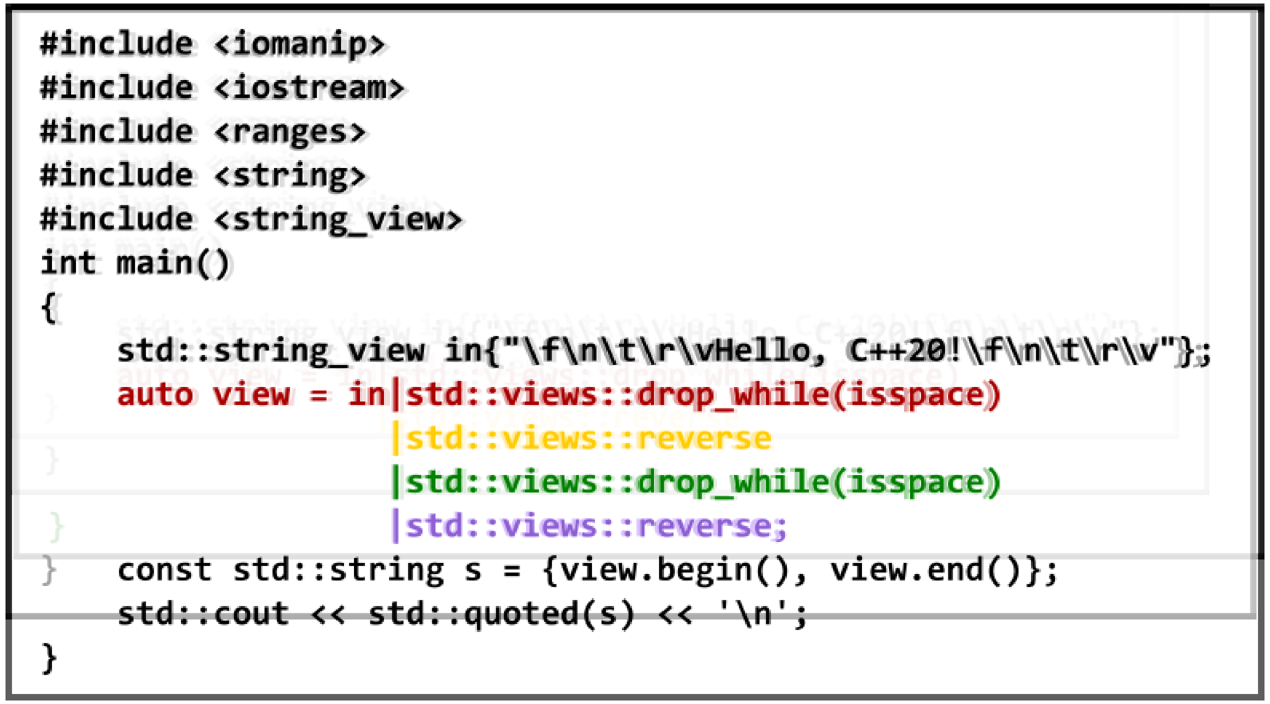
```cpp
auto print = [](auto const& view)
{
    for (std::cout <<'\n'<< "{ "; const auto element : view)
        std::cout << element << ' ';
    std::cout << "} ";
};
```

Strings: Removing white space from a String

\f\n\t\r\vHello, C++20!\f\n\t\r\v	→	Hello, C++20!
\f\n\t\r\vHello, C++20!\f\n\t\r\v	→	Hello, C++20!\f\n\t\r\v
Hello, C++20!\f\n\t\r\v	→	\v\r\t\n\f!02++C, olleH
\v\r\t\n\f!02++C, olleH	→	!02++C, olleH
!02++C, olleH	→	Hello, C++20!

Strings: Removing white space from a String

- ✓ Remove all till we found first non white space character.
- ✓ Reverse the view
- ✓ Remove all till we found first non white space character.
- ✓ Reverse the view
- ✓ Print the results in quotes

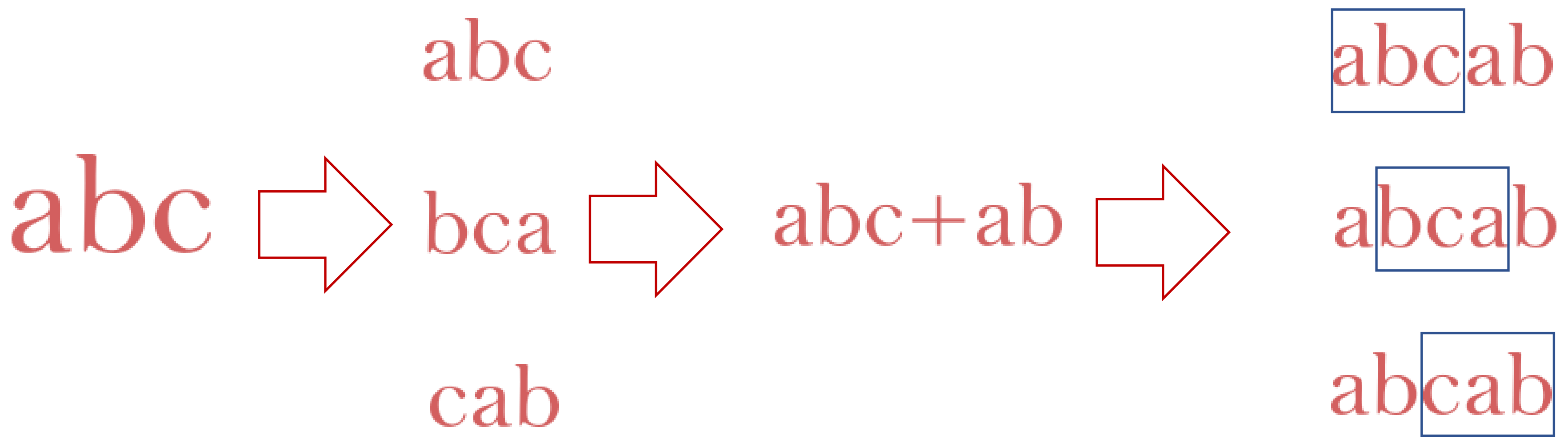


```
#include <iomanip>
#include <iostream>
#include <ranges>
#include <string>
#include <string_view>
int main()
{
    std::string_view in{"\f\n\t\r\vHello, C++20!\f\n\t\r\v"};
    auto view = in|std::views::drop_while(isspace)
                  |std::views::reverse
                  |std::views::drop_while(isspace)
                  |std::views::reverse;
    const std::string s = {view.begin(), view.end()};
    std::cout << std::quoted(s) << '\n';
}
```

“Hello, C++20!”

Strings : All Rotations of a String

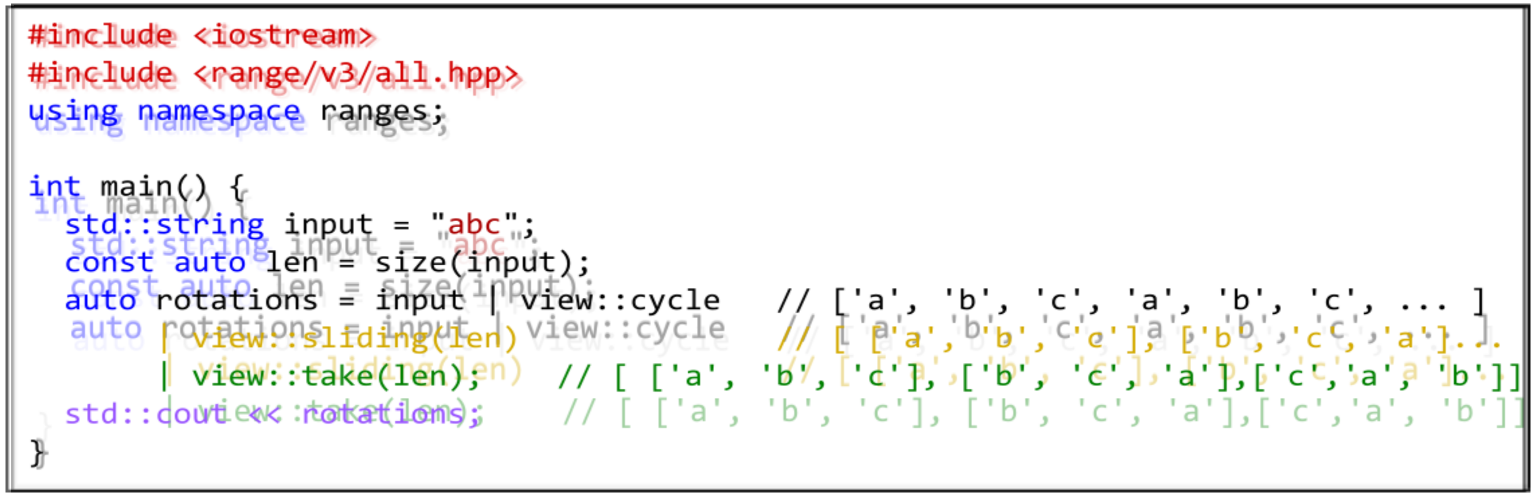
Thanks To **Marco Arena**



- ✓ Every rotation is a substring of the original string concatenated with itself (last character excluded).
- ✓ The number of rotations is equal to the length of the string.

Strings : All Rotations of a String

- ✓ Setup and Create Cyclic view.
- ✓ Create Sliding view.
- ✓ Take the number first(length).
- ✓ Print all rotations



```
#include <iostream>
#include <range/v3/all.hpp>
using namespace ranges;

int main() {
  std::string input = "abc";
  const auto len = size(input);
  auto rotations = input | view::cycle   // ['a', 'b', 'c', 'a', 'b', 'c', ... ]
    | view::sliding(len)                 // [ ['a', 'b', 'c'], ['b', 'c', 'a'], ['c', 'a'] ...
    | view::take(len);                   // [ ['a', 'b', 'c'], ['b', 'c', 'a'],['c','a', 'b']]
  std::cout << rotations;                // [ ['a', 'b', 'c'], ['b', 'c', 'a'],['c','a', 'b']]
}
```

abc bca cab

Thank You

+91-9886772742

harishankar.singh@qickstep.in

www.qickstep.in

www.linkedin.com/in/harishankarsinghyadav