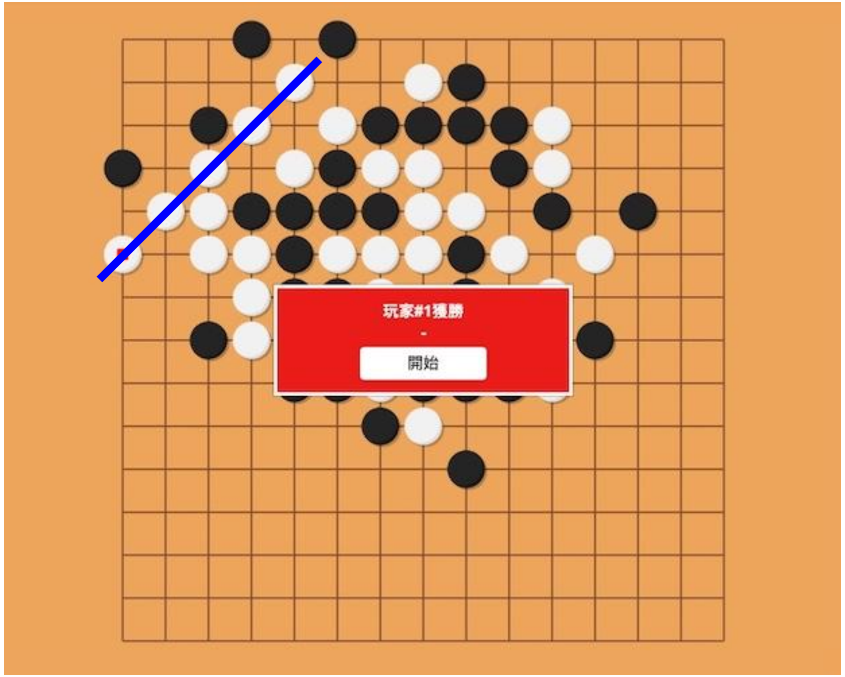
Compile Time Game- Gomoku

Shao-Chi Wu (sw3525)
Wei-Ren Lai   (wl2777)
Yen-Min Hsu (yh3328)

Instructor: Prof. Bjarne Stroustrup



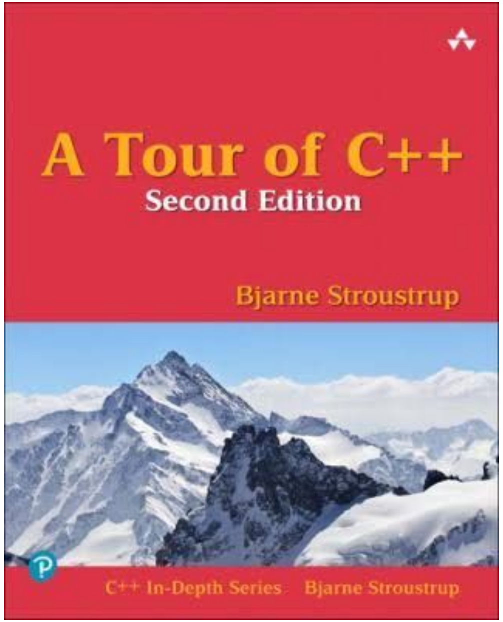
玩家#1獲勝

-

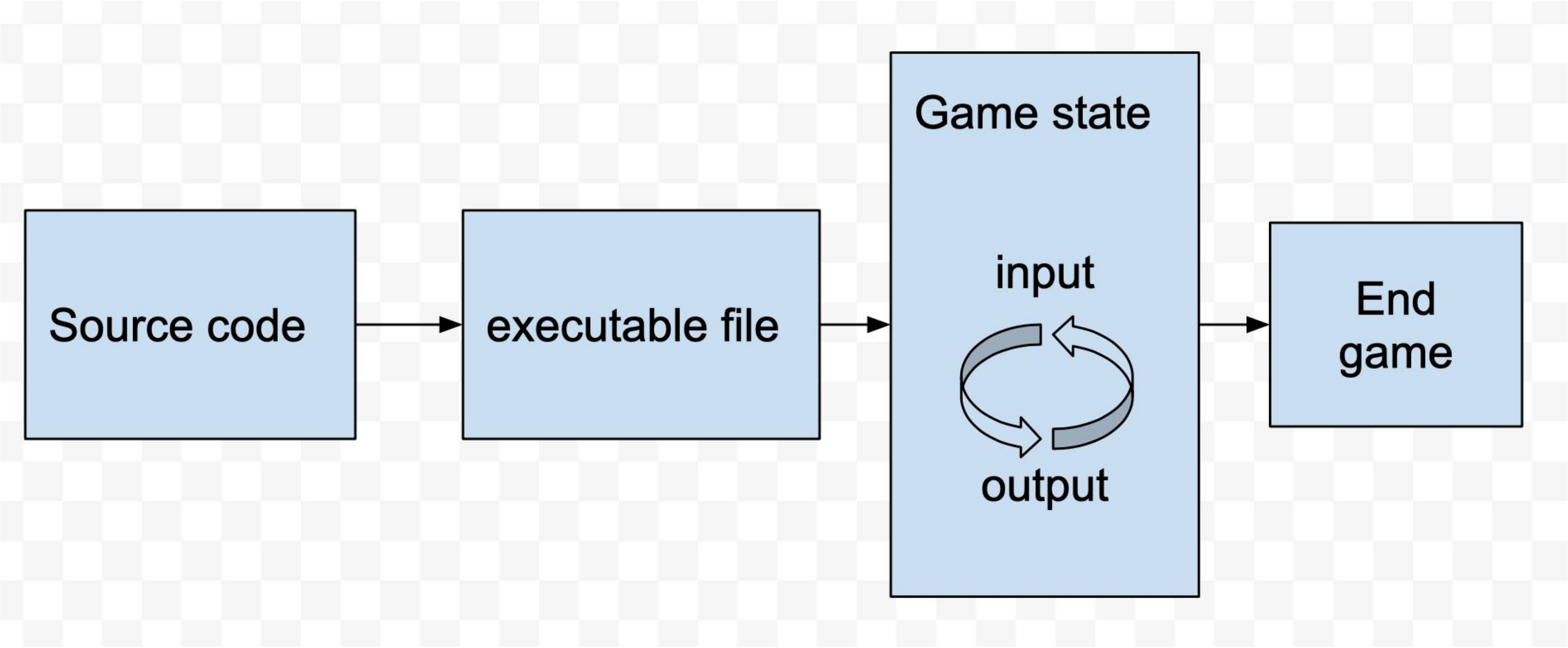
Motivation

- C++
  - Compile-time Implementations
  - Generic Programming
  - C++ evolution in C++17/20

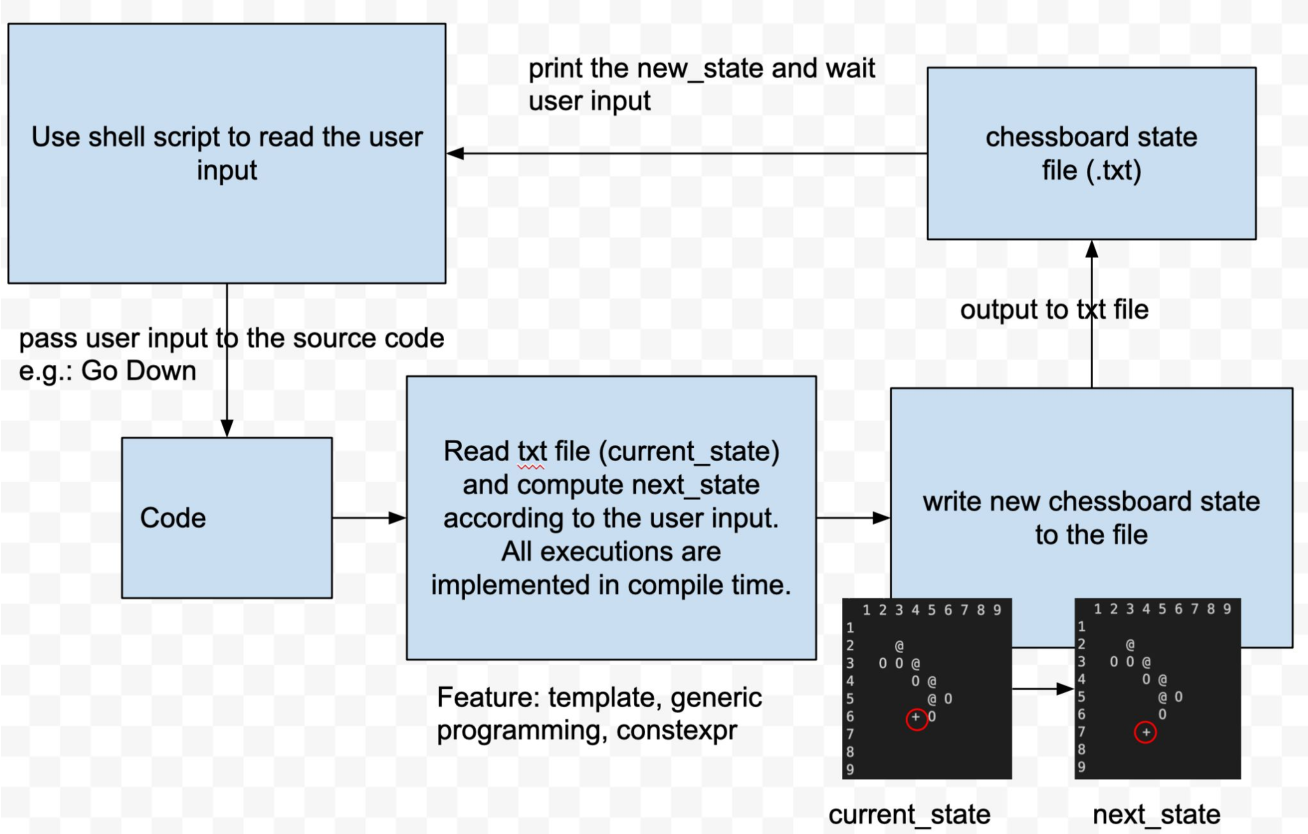
- Final Project Guideline
  - Interesting
  - Interactive



Design a game (run-time version)



Design a **compile time** game



Use shell script to read the user input
print the new_state and wait user input
chessboard state file (.txt)
pass user input to the source code e.g.: Go Down
Code
Read txt file (current_state) and compute next_state according to the user input. All executions are implemented in compile time.
write new chessboard state to the file
output to txt file
Feature: template, generic programming, constexpr

```
  1 2 3 4 5 6 7 8 9
1
2     @
3   0 0 @
4       0 @
5         @ 0
6       + 0
7
8
9
```

current_state

```
  1 2 3 4 5 6 7 8 9
1
2     @
3   0 0 @
4       0 @
5         @ 0
6         0
7       +
8
9
```

next_state

Spec

- The program can run end-to-end without error.
- The program can terminate when a draw situation or one of the players wins.
- When the cursor(+) attempt go out of the board, it stays at the same place.
- When the player select the same grid to place the stone, this move would be ignored. Also, the player’s turn remains.

Implementation

Design Feature (C++17/20):

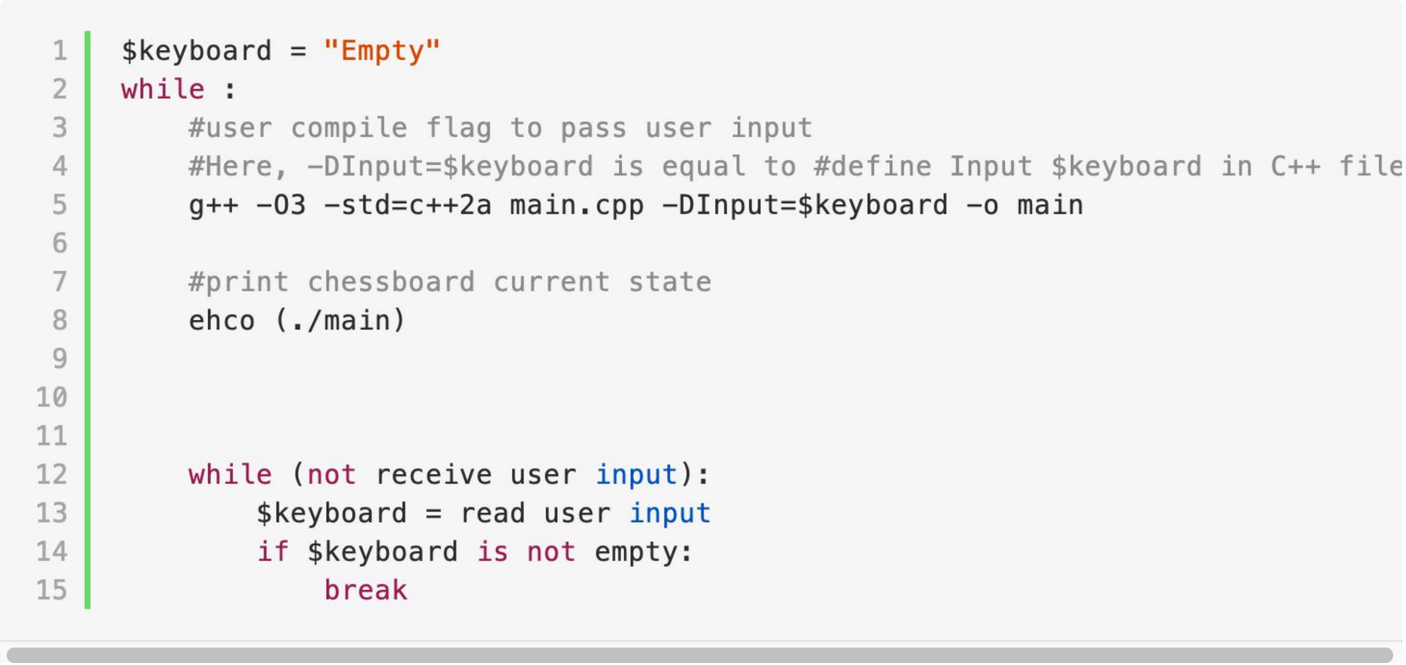
- template meta programming
- std::string_view
- std::from_chars_result from_chars (stoi for string_view)
- std::integer_sequence / std::index_sequence

Design Feature (Others):

- Write a shell scipt to maintain the game loop.

How to construct a compile-time game?
First we need to write a shell script to read user input

Here is the pseudo code:

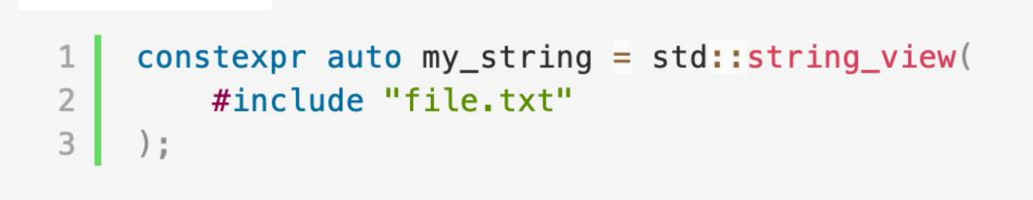


```
$keyboard = "Empty"
while :
    #user compile flag to pass user input
    #Here, -DInput=$keyboard is equal to #define Input $keyboard in C++ file
    g++ -O3 -std=c++2a main.cpp -DInput=$keyboard -o main

    #print chessboard current state
    ehco (./main)



    while (not receive user input):
        $keyboard = read user input
        if $keyboard is not empty:
            break
```

How to read file and save

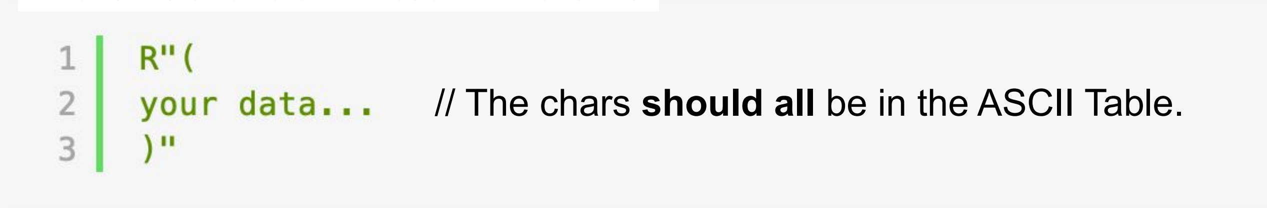
ANGELOU · ANZALDÚA · CHANG · HURSTON · MORRISON · REVATHI · SHANGE · SILKO
ER · HERODOTUS · SOPHOCLES · PLATO · ARISTOTLE · DEMOSTHENES · CICERO · VE

Get text file:



```
constexpr auto my_string = std::string_view(
    #include "file.txt"
);
```

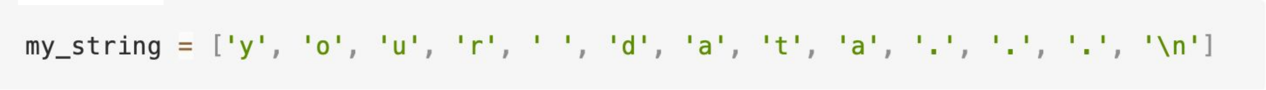
File should be written like this:



```
R"(
your data...
)"
```

// The chars **should all** be in the ASCII Table.

Result:



```
my_string = ['y', 'o', 'u', 'r', ' ', 'd', 'a', 't', 'a', '.', '.', '.', '\n']
```

User-defined string (class STR)

Why?

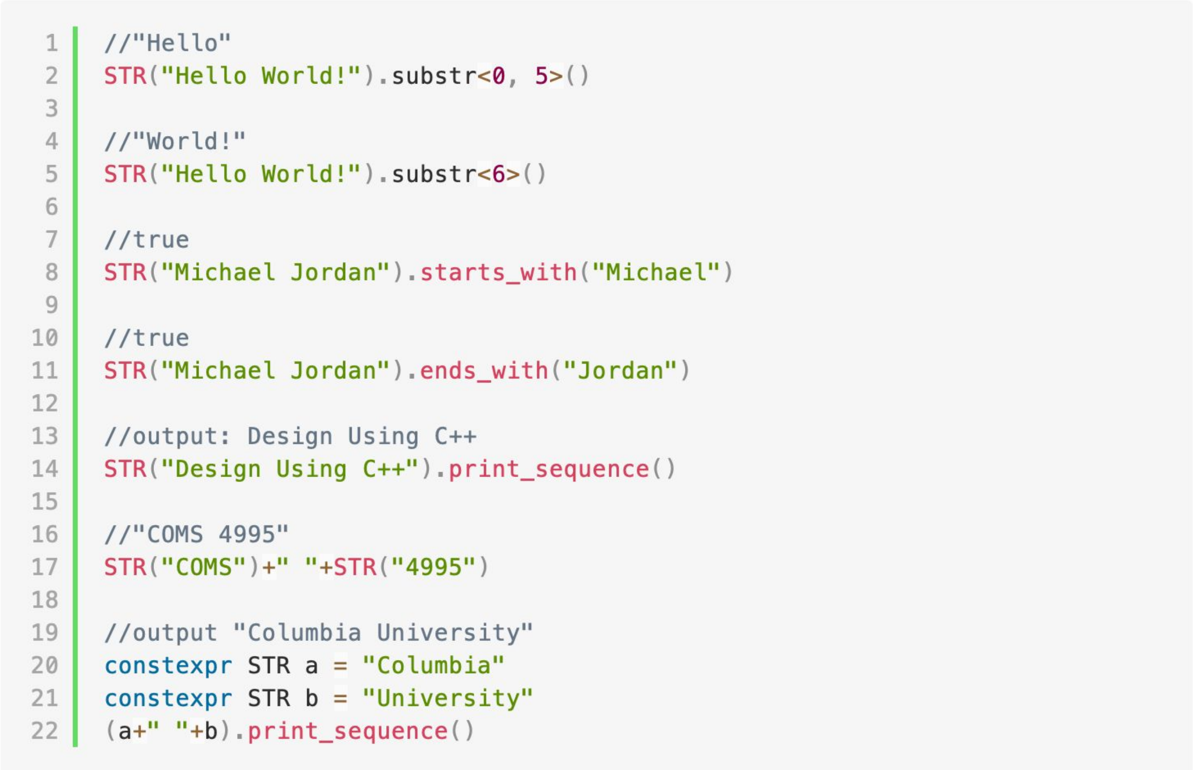
- You need a string class that can be implemented in compile-time, so that you can save the data from the txt file (read input and save)
- std::string is not yet be supported in compile-time by current compilers(C++20)
- std::string_view is a good choice, and it's faster too!
- However, we want to implement some functions that std::string_view doesn't provide (e.g.: concatenate multiple strings), so we design a new class

Some support functions in STR:

- substr (STR.substr<start, length>())
  - similar to string.substr(), but here we use template to pass the size of the new STR
- starts_with
  - check whether the string starts with the given string
- ends_with
  - similar to the function starts_with
- size(), length()
  - return current STR.size()
- operator+
  - support to concatenate STRS
- operator==
  - support to compare two STRS

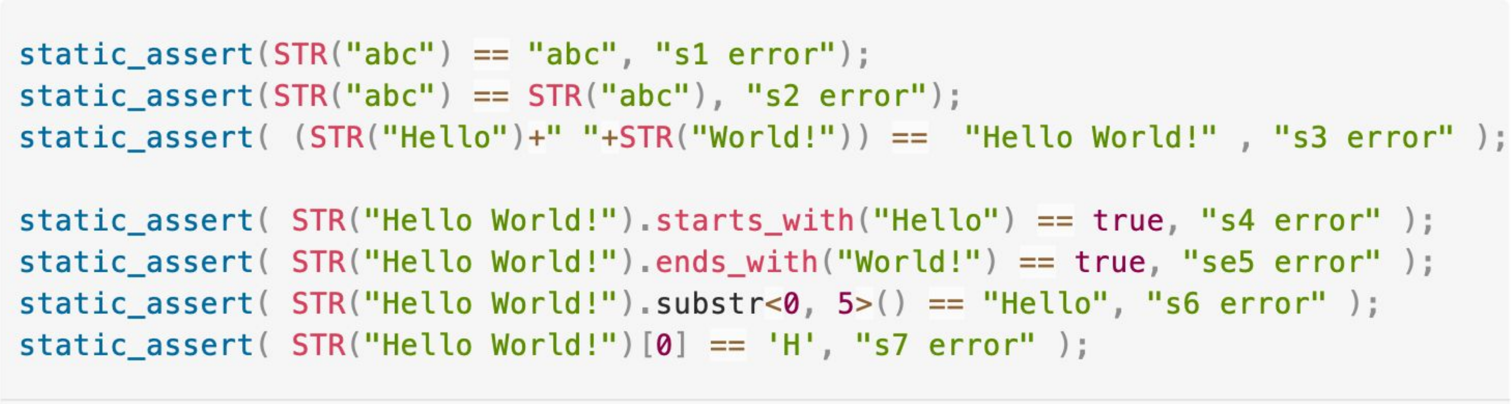
© Dado Ruvic/REUTERS

Sample implementations of our class STR



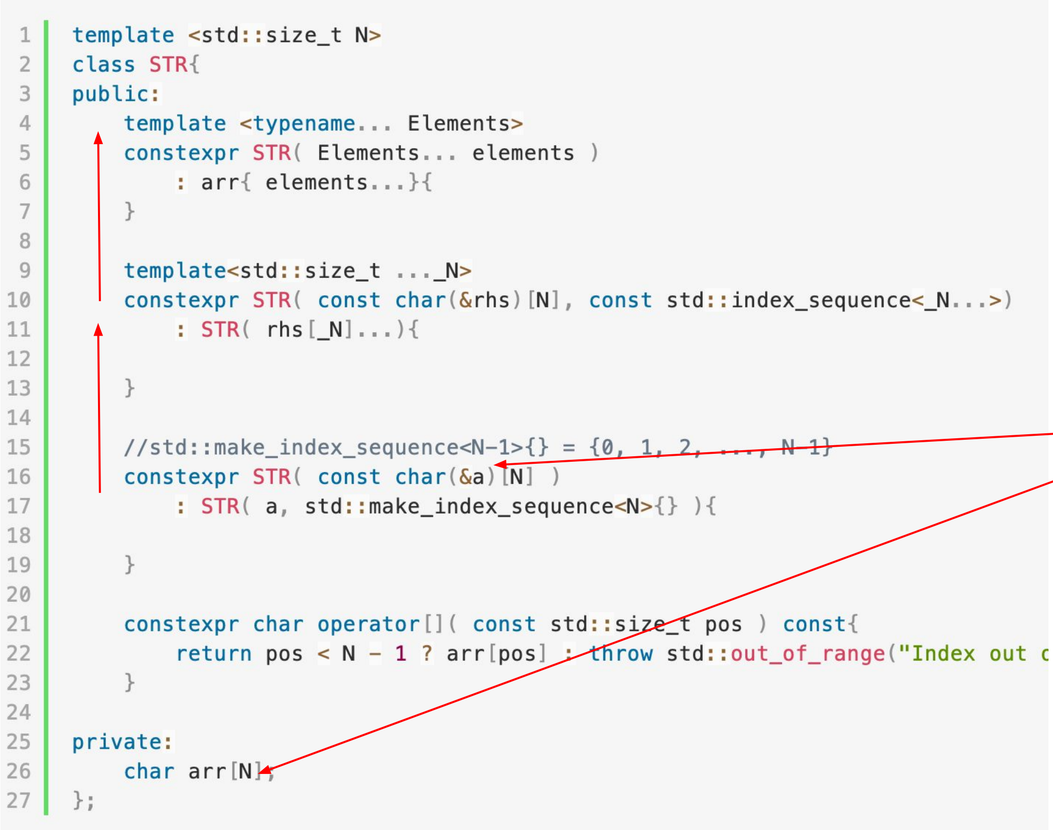
```
//"Hello"
STR("Hello World!").substr<0, 5>()

//"World!"
STR("Hello World!").substr<6>()

//true
STR("Michael Jordan").starts_with("Michael")

//true
STR("Michael Jordan").ends_with("Jordan")

//output: Design Using C++
STR("Design Using C++").print_sequence()

//"COMS 4995"
STR("COMS")+" "+STR("4995")

//output "Columbia University"
constexpr STR a = "Columbia"
constexpr STR b = "University"
(a+" "+b).print_sequence()
```

Use static_assert to test user-defined STR correctness:



```
static_assert(STR("abc") == "abc", "s1 error");
static_assert(STR("abc") == STR("abc"), "s2 error");
static_assert( (STR("Hello")+" "+STR("World!")) ==  "Hello World!" , "s3 error" );

static_assert( STR("Hello World!").starts_with("Hello") == true, "s4 error" );
static_assert( STR("Hello World!").ends_with("World!") == true, "se5 error" );
static_assert( STR("Hello World!").substr<0, 5>() == "Hello", "s6 error" );
static_assert( STR("Hello World!")[0] == 'H', "s7 error" );
```

How we construct the class STR (using template)



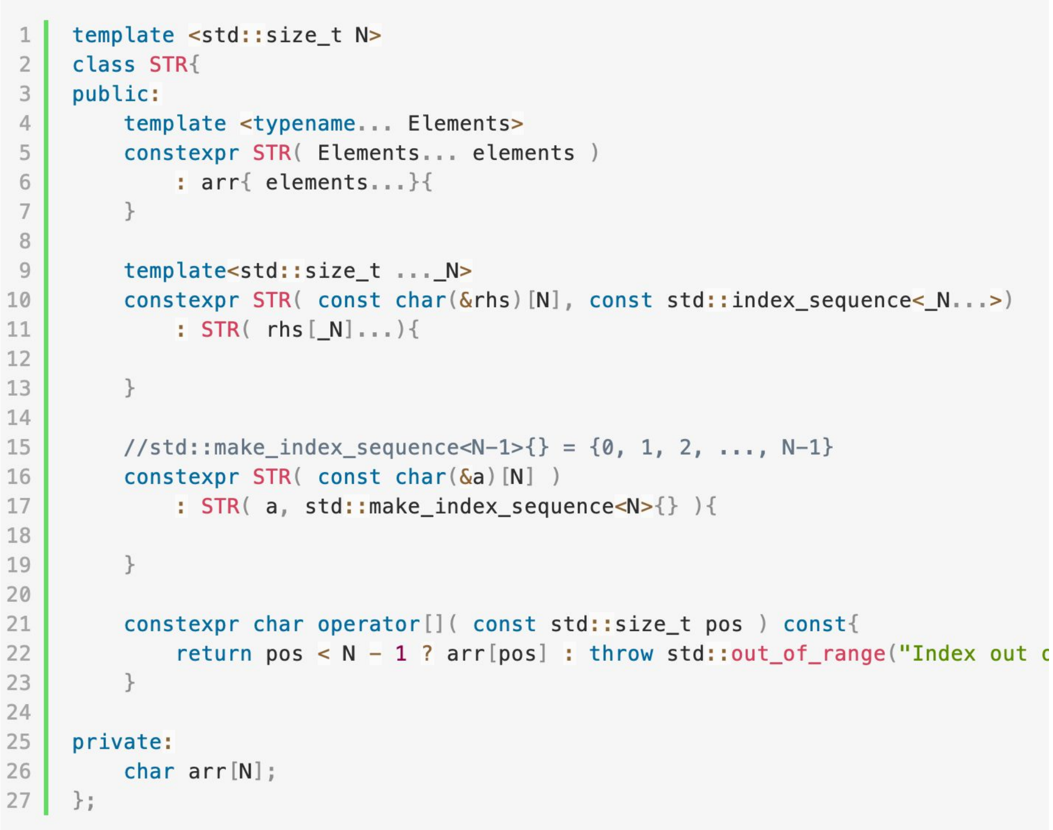
```
template <std::size_t N>
class STR{
public:
    template <typename... Elements>
    constexpr STR( Elements... elements )
        : arr{ elements...}{
    }

    template<std::size_t ..._N>
    constexpr STR( const char(&rhs)[N], const std::index_sequence<_N...>)
        : STR( rhs[_N]...){

    }

    //std::make_index_sequence<N-1>{} = {0, 1, 2, ..., N-1}
    constexpr STR( const char(&a)[N] )
        : STR( a, std::make_index_sequence<N>{} ){

    }

    constexpr char operator[]( const std::size_t pos ) const{
        return pos < N - 1 ? arr[pos] : throw std::out_of_range("Index out 
    }

private:
    char arr[N];
};
```

Note:
std::make_index_sequence<10>{}
= {0, 1, 2, …, 9}

In brief, the constructor is trying to do:

```
for(size_t i=0; i<N; ++i)
        arr[i] = a[i]
```

How we construct the class STR (using template)



```
template <std::size_t N>
class STR{
public:
    template <typename... Elements>
    constexpr STR( Elements... elements )
        : arr{ elements...}{
    }

    template<std::size_t ..._N>
    constexpr STR( const char(&rhs)[N], const std::index_sequence<_N...>)
        : STR( rhs[_N]...){

    }

    //std::make_index_sequence<N-1>{} = {0, 1, 2, ..., N-1}
    constexpr STR( const char(&a)[N] )
        : STR( a, std::make_index_sequence<N>{} ){

    }

    constexpr char operator[]( const std::size_t pos ) const{
        return pos < N - 1 ? arr[pos] : throw std::out_of_range("Index out o
    }

private:
    char arr[N];
};
```

For example:
constexpr char a[] = “test”;
constexpr STR str = a;

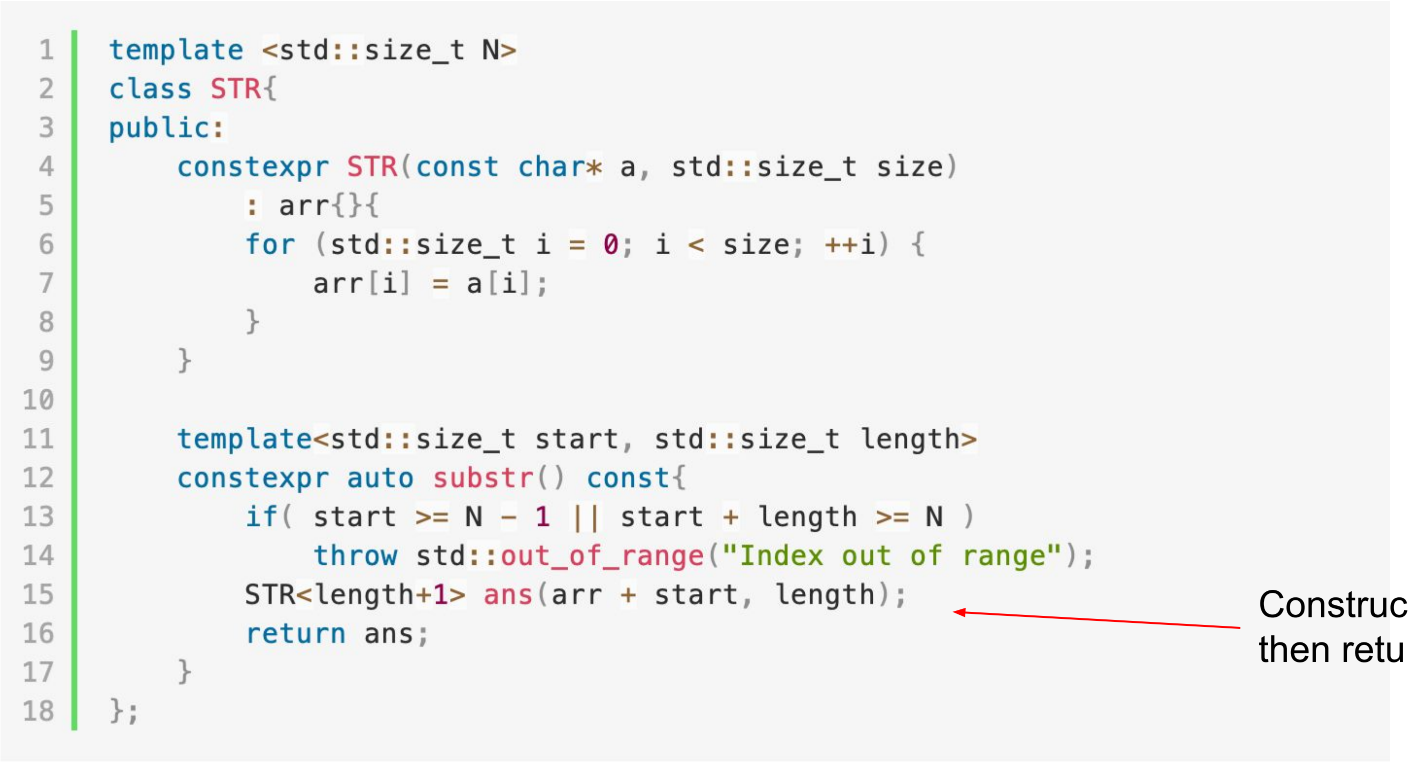
Now, N = 5

std::make_index_sequence<5>{} = {0, 1, 2, 3, 4}

```
for(size_t i=0; i<5; ++i)
        arr[i] = a[i]
```

-> arr[N] = [‘t’, ‘e’, ‘s’, ‘t’, ‘\0’]

Sample implementation of substr() function



```
template <std::size_t N>
class STR{
public:
    constexpr STR(const char* a, std::size_t size)
        : arr{}{
        for (std::size_t i = 0; i < size; ++i) {
            arr[i] = a[i];
        }
    }

    template<std::size_t start, std::size_t length>
    constexpr auto substr() const{
        if( start >= N - 1 || start + length >= N )
            throw std::out_of_range("Index out of range");
        STR<length+1> ans(arr + start, length);
        return ans;
    }
};
```

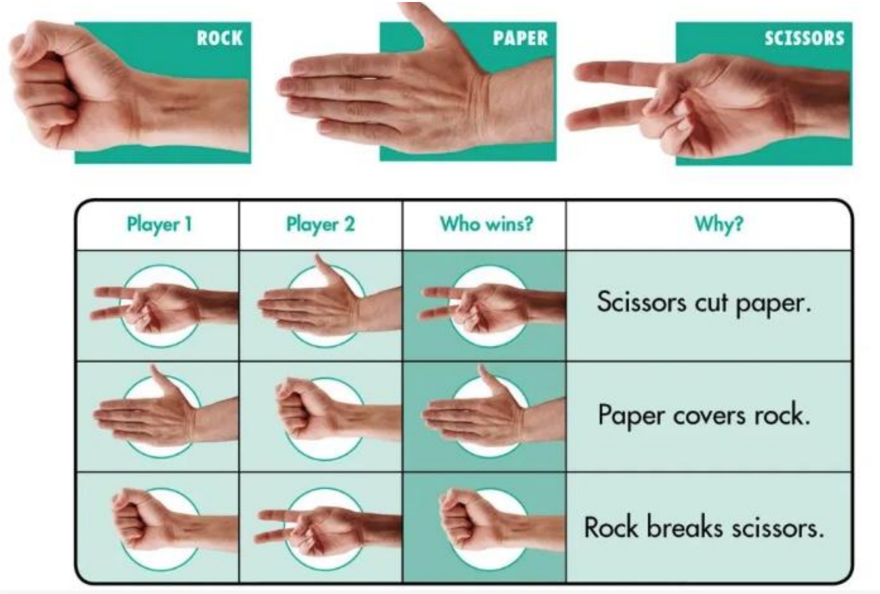
Construct a new STR and then return

Tutorial

How to start building your own compile time game? 🤔

Let’s find our childhood momories!

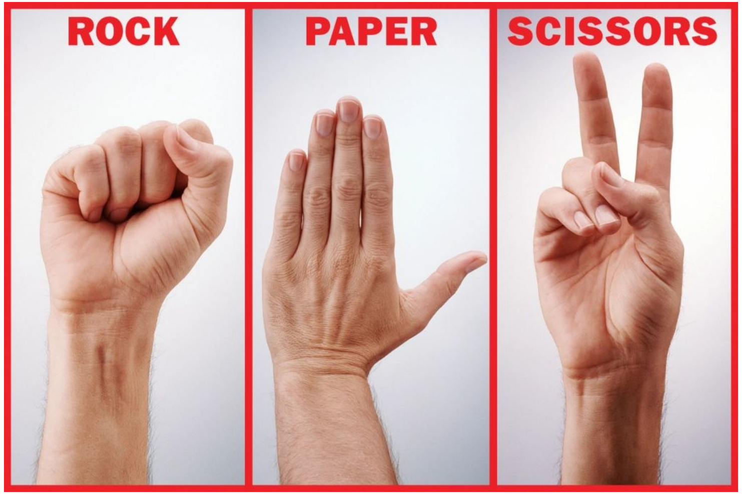
- A simple tutorial - Rock paper scissors
- Want more? See the tutorial section on GitHub



Scissors cut paper.

Paper covers rock.

Rock breaks scissors.



re: https://www.timeforkids.com/k1/rock-paper-scissors/

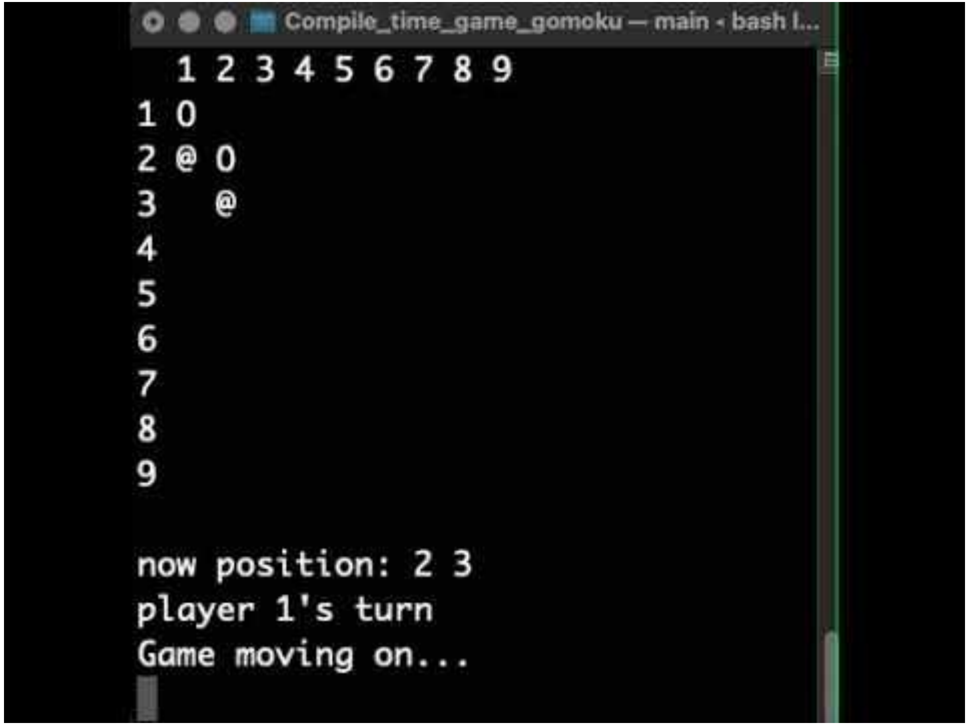
Tutorial

Step 0. Design your game states

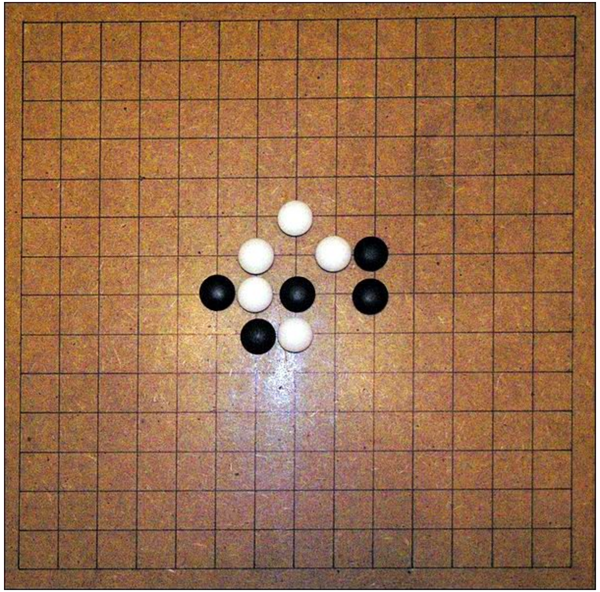
Step 1. Assume user input and integrate

Step 2. Use shell script to connect the modules

DEMO



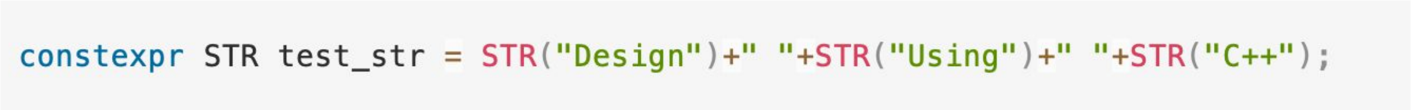
```
  1 2 3 4 5 6 7 8 9
1 O
2 @ O
3   @
4
5
6
7
8
9

now position: 2 3
player 1's turn
Game moving on...
```



re: https://en.wikipedia.org/wiki/Gomoku

Mesurement

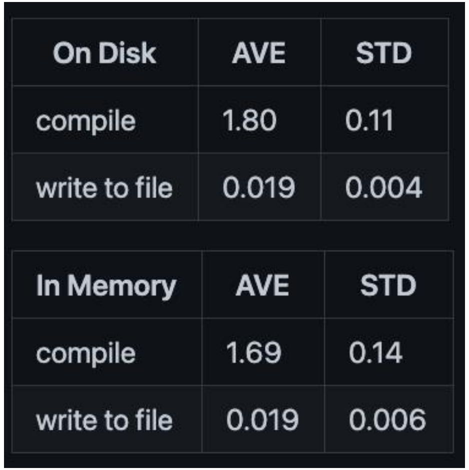
- Average compile time per step: 0.885 sec.
- User-defined STR vs std::string_view
  - Initialize a very long string: string_view is 4 times faster.
  - Get the sub-string: string_view is 3 times faster.
  - Still, we need STR… (supporting "+" operator)



```
constexpr STR test_str = STR("Design")+" "+STR("Using")+" "+STR("C++");
```

Mesurement

- In-Memory vs On-Disk Game States
  - The difference between the two scenarios is not significant.
  - Our game states are only hundreds of bytes.

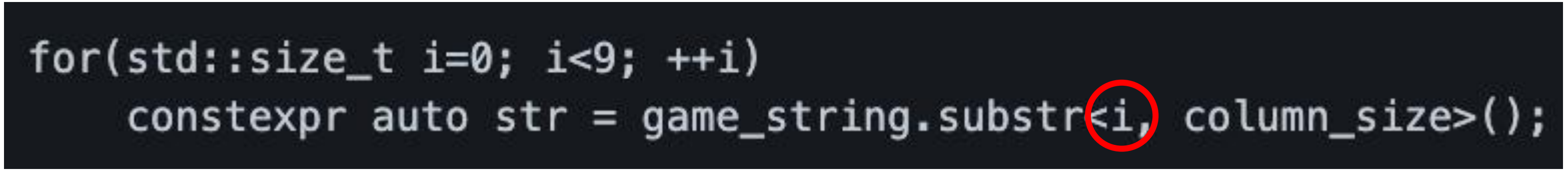


On Disk	AVE	STD
compile	1.80	0.11
write to file	0.019	0.004

In Memory	AVE	STD
compile	1.69	0.14
write to file	0.019	0.006

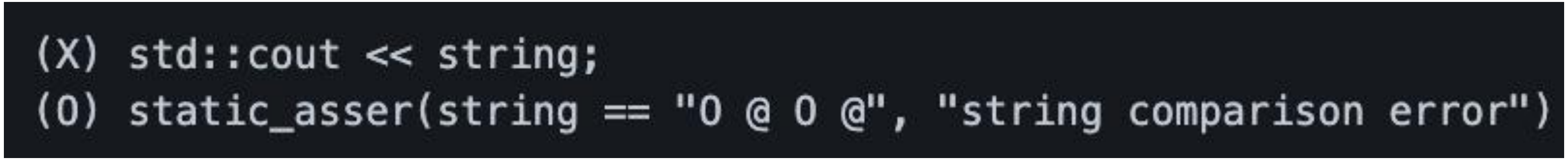
Bad attempts we made

- Using in-memory file systems or memory-mapped file.
- Pass non const/constexpr element into template.



```
for(std::size_t i=0; i<9; ++i)
    constexpr auto str = game_string.substr<i, column_size>();
```

- Using std::cout to debug.



```
(X) std::cout << string;
(O) static_asser(string == "O @ O @", "string comparison error")
```

Future Work

- Implement an AI player, so that a user can play with the computer.
    - alpha-beta pruning or similar algorithms

- Model-view-controller (MVC) design pattern

Take away

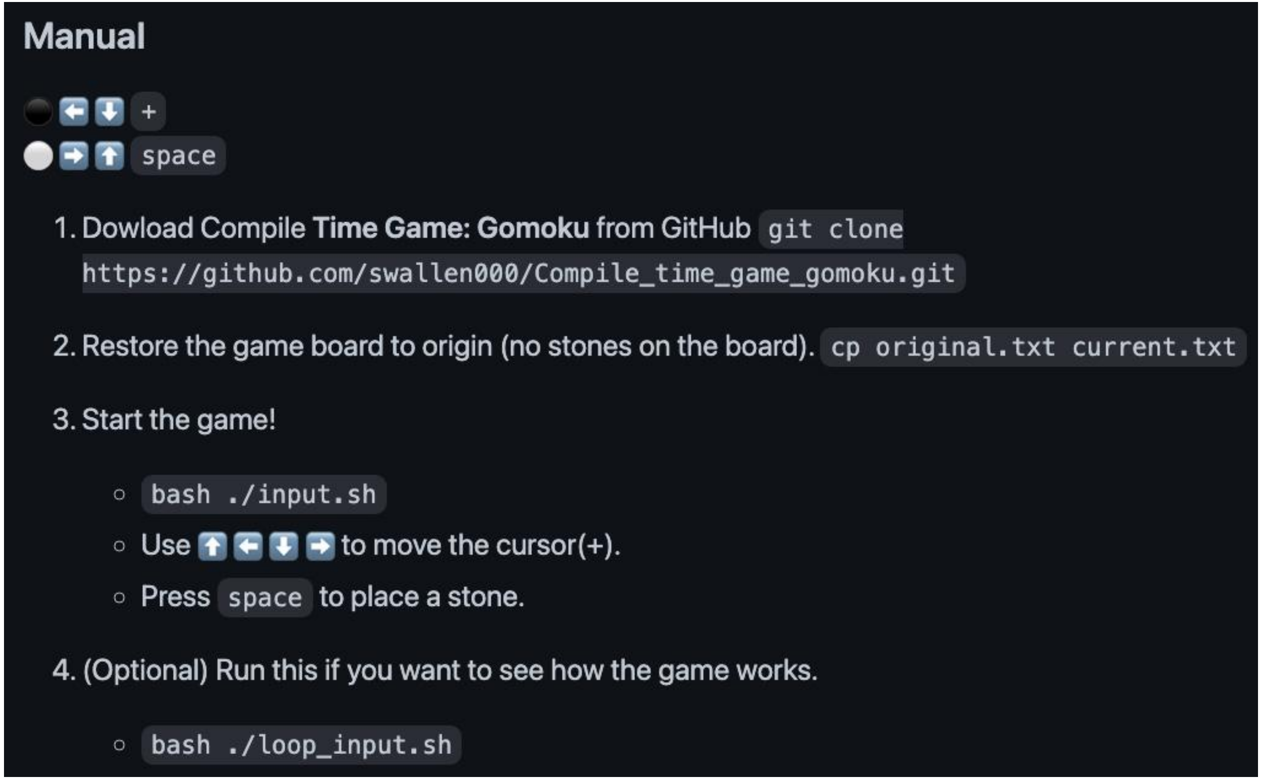
- Compile game has a different design logic (e.g. inputs, game states)
- C++ is very powerful in compile-time implementations
- template is very powerful for generic programming
- using static_assert to help debug when trying to write functions in compile-time
- template mega recursions have limited depth, we can use compile flag “-ftemplate-depth=” to set required depth. However, due to the compiler limitation and hardware limitation, we can't set the value arbitrarily high as we want.
- We should not expect a compile time game is more efficient than its runtime version. However, it is good that compile time game finds error earlier than its runtime version.

Reference

- Error spliting an std::index_sequence
    - stackoverflow.com/questions/20874388/error-spliting-an-stdindex-sequence
- Compile-time strings and string concatenation
    - https://gist.github.com/dominicusin/b4008ab9895240f615be6a886eb81829
- dsanders11/StringConstant.h
    - https://gist.github.com/dsanders11/8951887
- std::basic_string_view
    - https://en.cppreference.com/w/cpp/string/basic_string_view
- std::integer_sequence
    - https://en.cppreference.com/w/cpp/utility/integer_sequence
- Jiwan/meta_crush_saga
    - https://github.com/Jiwan/meta_crush_saga

Try it!

https://github.com/swallen000/Compile_time_game_gomoku



Manual

● ⬅ ⬇ `+`
○ ➡ ⬆ `space`

1. Dowload Compile **Time Game: Gomoku** from GitHub `git clone https://github.com/swallen000/Compile_time_game_gomoku.git`
2. Restore the game board to origin (no stones on the board). `cp original.txt current.txt`
3. Start the game!
   - `bash ./input.sh`
   - Use ⬆ ⬅ ⬇ ➡ to move the cursor(+).
   - Press `space` to place a stone.
4. (Optional) Run this if you want to see how the game works.
   - `bash ./loop_input.sh`

Q & A

or email: {sw3525, wl2777, yh3328} @columbia.edu