4 Template class/function and Separate Compilation

We can separate CyclicArray class into a header file and a implementation file like:

- **cyclicarray.h**
- **cyclicarray.cpp**
- **main.cpp**

However, this does **not work!**

When a application is built from multiple ***.cpp** files, each ***.cpp** file is compiled one by one then the linker links those together.

- **cyclicarray.cpp →cyclicarray.o**
- **main.cpp →main.o**
- **cyclicarray.o + main.o → a.out**

The problem occurs when the linker links object files.

```
Undefined symbols for architecture x86_64:
  "CyclicArray<int, 16>::CyclicArray()", referenced from:
      _main in main.o
  "CyclicArray<int, 16>::~CyclicArray()", referenced from:
      _main in main.o
  "CyclicArray<int, 16>::operator[](int)", referenced from:
      _main in main.o
ld: symbol(s) not found for architecture x86_64
clang: error: linker command failed with exit code 1 (use -v to see invocation)
```

- The linker couldn't find a specific type and value of definition in the object file.
- When the compiling **cyclicarray.cpp →cyclicarray.o**, type and value are not determined.
- **main.cpp** determines the type and value for CyclicArray template class.
- We cannot compile **main.cpp** and **cyclicarray.cpp** separately.

What to do?

We can include **cyclicarray.cpp** into **main.cpp** or **cyclicarray.h**, so that **CyclicArray** class implementation is compiled with **main.cpp**

4.1 Implement class in header file

Usually, header files for template class include its implementation. So no *.cpp file!

```cpp
#include <iostream>
template<class T, int SIZE>
class CyclicArray{
public:
    // default constructor
    CyclicArray()
    {
      // nothing to do
    }

    // destructor
    ~CyclicArray()
    {
      // nothing to do
    }

    // [] operator overload Note that it returns a reference
    T &operator[](int i)
    {
        // Cyclic mechanism
        i = i % SIZE;
        if (i < 0) i += SIZE;
        return array[i];
    }

private:
    // integer array
    T array[SIZE];
};
```