CPP-05

* Generic programming
  → Separate algorithms from the data type.

⇒ Generic programming uses Keyword template:

Example

```
template <typename T, typename S>
T function (Const T& Var_t, Const S& Var_s){
    T result = Var_t;
    return result;
}
```

⇒ Templates are used for meta programming

⇒ If we create MyClass<int> and MyClass<float> the compiler will generate two different classes with appropriate type instead of template parameter.

⇒ Declare in NAME.h file, implement in NAME.hpp file, add #include <NAME.hpp> in the end of NAME.h.

* Iterators

↳ STL uses iterators to access data in Containers.

* Error handling with exceptions

⇒ To use exceptions use #include <stdexcept>

* throw exception

→ Runtime Error

```
if (badEvent) {
    string msg = "specific error string";
    throw runtime_error (msg);
}
```

→ Logic Error

```
throw logic_error (msg);
{an error in logic of the user}
```

* Catch exceptions

⇒ If we expect an exception, we can "catch" it.

⇒ Use try-catch to catch exceptio

```
try {
  x = SomeUnsafeFunction(a, b, c)
}
catch (runtime_error &ex) {
  cout << ex.what();
}
catch (logic_error &ex) {
  cout << ex.what();
}
Catch (...) { //all other
  cout << "Unknown";
}
```

⇒ GOOGLE-STYLE ⟶ Don't use exceptions

* Program input parameters

⇒ Allow passing arguments to the binary.

⇒ int main (int argc , char const* argv[]);

⇒ argc ⇒ defines number of input parameters

⇒ argv ⇒ is an array of string parameters.

default

argc == 1

argv = "<binary_path>"

* Using for type aliasing

'Using NewType = OldType;'

⇒ Type is available in the Scope of declaration.