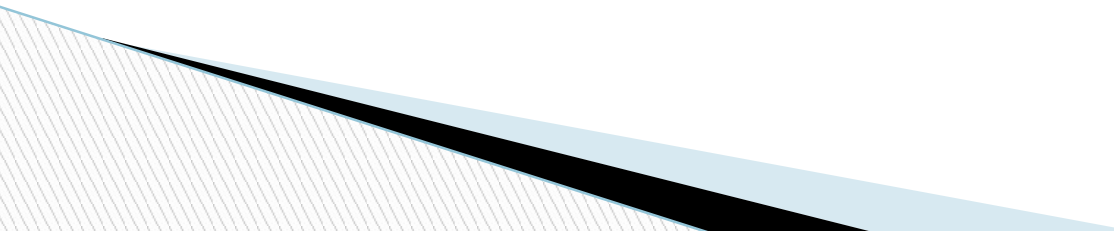
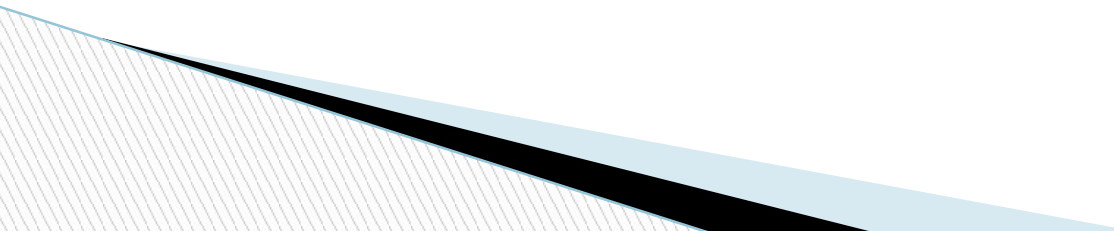


# Namespace

- Namespace is a declarative region providing a scope to the identifiers contained in it.
  - scope or visibility : **Local namespace and Global namespace**
  - **Unique name**
- 

# Need of namespace

- ❑ Real applications are usually very large and divided into multiple modules or tasks.
  - ❑ These modules are mostly developed by more than one developer in an independent manner.
  - ❑ To produce the final application, these separate source files are compiled and linked.
  - ❑ **Namespaces** are used to organize code into logical groups and prevent name collisions that can occur especially when the code base includes multiple libraries.
- 

# Using namespace

- Using namespace involves:
  - Creating namespace with unique identifiers as its members or elements
  - Accessing elements or members of namespace

```
namespace namespace_name
{
    //namespace body
    // identifier declarations
}
```

# Importing namespace elements with keyword 'using'

- `using namespace <namespace_name>`
- Anonymous Namespace : It is possible to create a namespace without giving any name. Such namespaces are called anonymous or **unnamed namespaces**
- `namespace // anonymous, not given any name`
- `{ int x; int y; }`

# Nested namespaces

```
#include <iostream>

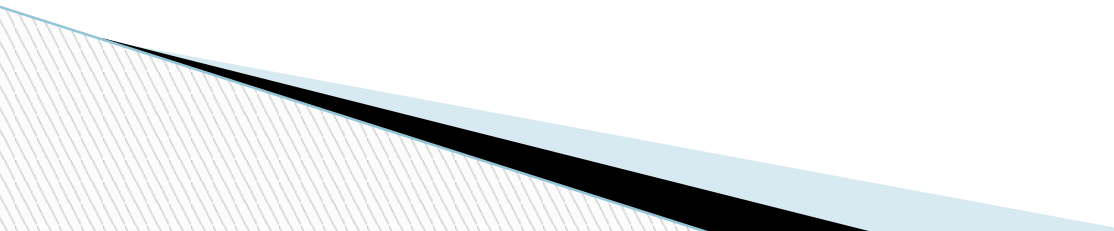
int x = 20;
namespace outer {
    int x = 10;
    namespace inner {
        int z = x; }
}

int main()
{
    std::cout<<outer::inner::z; //prints 10
    getchar();
    return 0;
}
```

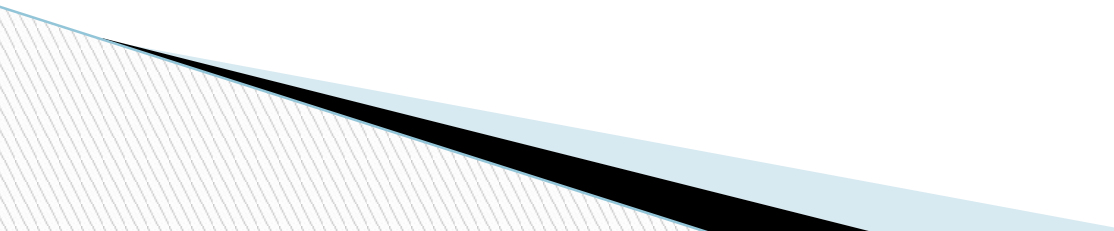
# Namespace Aliasing

- Namespace aliasing allow the programmer to define an alternate name for a namespace.
- `namespace new_name = current_name;`
- **Extending Namespace**
  - `namespace myNamespace { float num1 = 10.5; }`
  - `namespace myNamespace // extended myNamespace`
  - `{ float num2 = 30.75; }`

# Do namespace introduce any overhead?

- ❑ Namespace does not introduce any overhead at runtime as namespaces are resolved at compile time. All fully qualified name and using directive or declarations are handled at compile time.
  - ❑ Use of namespace does affect the readability of the program. So it should be used only when needed.
- 

# The Koenig Look Up

- Andrew Koenig
  - The Formal name of the algorithm is augment dependent lookup
  - The Koenig Look up is automatically applied by ANSI/ISO standard Compilers.
- 



# The Koenig Look Up

```
//KoenigLookup.cpp
#include <iostream>
// using namespace std is missing here
int main()
{
    using std::cout;
    // The following line is now commented // using std::operator <<;
    cout << "Hi";
    return 0;
}
```

The program still works as before. The operator << will be found from the std namespace by the Koenig lookup algorithm.