Namespaces

Overview

An Introduction to namespaces
Defining namespaces
Using namespaces

2

An Introduction to Namespaces

Allow an application to be partitioned into a number of subsystems
Each subsystem can define entities and operate in its own scope
No need to worry about identifiers used by someone else

3

Why use Namespaces?

Solve problem of name clashes in the global scope
Correspond to subsystems from RD/OOA
Can concentrate on local issues without worrying about other subsystems

4

Defining Namespaces

Namespace is defined by using the keyword `namespace`

```
namespace DATASIM
{
  double pi = 3.141592653;

  double func(double x)
  {
    return x * 2.0;
  }
}
```

5

Defining Namespace Rules

A namespace must be appear at file-scope

A namespace may also be defined inside another namespace (nesting)

A definition can be split up

6

Nesting Namespaces

```
namespace DATASIM
{
  double pi = 3.141592653;

  namespace Inner
  {
    double Distance = 123;
  }
}
```

7

Splitting Namespaces

```
namespace DATASIM
{
  double pi = 3.141592653;
}

void foo(){}

// The rest of the namespace, it is split
namespace DATASIM
{
  double func(double x)
  {
    return x * 2.0;
  }
}
```

8

Defining Outside Namespace

```
namespace DATASIM
{
  double pi = 3.141592653;
  double func(double);
}

void foo(){}

// The rest of the namespace, it is split
double DATASIM::func(double x)
{
  return x * 2.0;
}
```

9

Accessing Elements of a Namespace

By the using declaration
By the using directive
Explicit qualification
Identifiers in global namespace still accessible by using '::'

10

Using Declaration

The using declaration introduces a name into the declarative region

```
void main()
{
  // Employing the using declaration
  using DATASIM::func;

  cout << func(3.0);    // Gives value 6.0
}
```

11

Using Directive

The using directive allows **all** the names in a namespace to be used without qualification

```
void main()
{
  // Employing the using directive
  using namespace DATASIM;

  cout << func(3.0);    // Gives value 6.0
  cout << pi << endl;
}
```

12

Explicit Access Qualification

Identify namespace with each member usage

```
void main()
{
  cout << DATASIM::func(3.0);    // Gives value 6.0
}
```

13

Alias Namespaces

Create alias for namespace identifier

Possible to create shorter notation for nested aliases

```
namespace DS = DATASIM;          // Alias
cout << DS::func(3.0);           // Calls DATASIM::func()

namespace DSI = DATASIM::Inner;  // Nested Alias
```

14