C++

- Basic concepts
- Inheritance
- Generic Programming
- The Standard Template Library
- Examples

F. Rebufat

Basic concepts

- **Object Oriented programming :**

Programmer (concept)

OBJECT

Shape Action

C++ translation

CLASS

Member datas type Methods

Instance of class

INSTANCE

C++ Class

```
class My-name {
  data members
    int dm1;
    double dm2;

  member functions
   My-name();
   My-name(int i , double x)
       {dm1 = i ; dm2 = x;}
   int get-member1() {return this.dm1;}
   void set-member1(int i) {dm1 = i;}
   ⋮
};
```

```
main() {
  My-name obj1;
  My-name obj2(10 , 5.034);
  obj1.set-member1(3);
  int value = obj1.get-member1();
}
```

Inheritance

TRIANGULATION

↓

DELAUNAY

↓

CONSTRAINED

Derived class inherits methods and data members from its parents

Derived class can overload methods

C++ supports multiple inheritance

Resolution and scope

use operator :: to say to what a method or a type belongs

C++ Special aspects

- Reference Parameters

```
void foo(int &count)
void foo(const int &count)
```

- Constructors and Destructors

objectname(...)

~objectname()

new *and* *delete*

- Public, Protected, Private

	Public	Protected	private
Everybody	Yes		
Derived class	Yes	Yes	
Member functions	Yes	Yes	Yes
Freinds	Yes	Yes	Yes

INRIA SOPHIA ANTIPOLIS

F. Rebufat

Generic Programming

Use of Template to allow a type
to be a parameter in the definition
of a class or a function

```
template<class C1>
class C2
{
  ....
};
```

Template parameters are instanciated
in the "main" program

⚠ Operations on template parameters
have to be implemented somewhere

→ Requirements are needed

Traits Class

A class that implement primitives and functionnalities needed by an object

→ Modularity
Flexibility
Genericity

Allows users defined objects that can be used with others CGAL classes

Minimal definition required for an object

- Types needed
- Basic construtors
- Predicates, elementary operations

Polygon Model

CGAL-POLYGON-2

Methods on polygons

insert, erase,reverse,area,size, access...

Traits class

Types : Number-type, point, vector, segment...

cross product, determinant, orientation, comparison...

Container for vertices :

Vertex = Traits::Point

INRIA SOPHIA ANTIPOLIS

F. Rebufat

STL
Standard Template Library

Standard library :

- Memory management, run-time info
- functions optimally implemented :
  sqrt, memmove...
- non primitive facilities :
  containers, sort functions, I/O streams, iterators

STL : Containers

- simple and efficient
- standard iterator
- homogeneous

INRIA SOPHIA ANTIPOLIS

F. Rebufat

Containers and Iterator

Sequences :
Vector, list, deque, stack, queue
Associative containers :
Map, set, hash set...
Requirement :
A copy method is needed for elements
Associative containers require that their elements can be ordered or compared

Iterators :
Standardized way to traverse a container
- Easy to use
- Efficient
- Standard

Function Objects :
Standardized way to apply functions on container elements
- Nonmodifying operations :
  for-each(), find(), count()...
- Modifying operations :
  transform(), replace(), remove(), reverse()...
- Sorting :
  sort(), merge(), partition()...
- Set operations :
  includes(), set-union(), set-intersection()...
- Min & Max :

INRIA SOPHIA ANTIPOLIS

F. Rebufat

A new traits class for polygon

A polygon traits class for graphic

We want to add graphic informations to geometric objects (points, segments) as color, size, width...

CGAL-Point-2 → My-Point-2

```
int size ;
int color ;
void draw() ;
```

CGAL-Segment-2 → My-Segment-2

```
int color;
int width ;
void draw() ;
```

My-Polygon-traits

```
typedef My-Point-2    Point-2;
typedef My-Segment-2 Segment-2;
```

→ Write somewhere how polygon has to be drawn