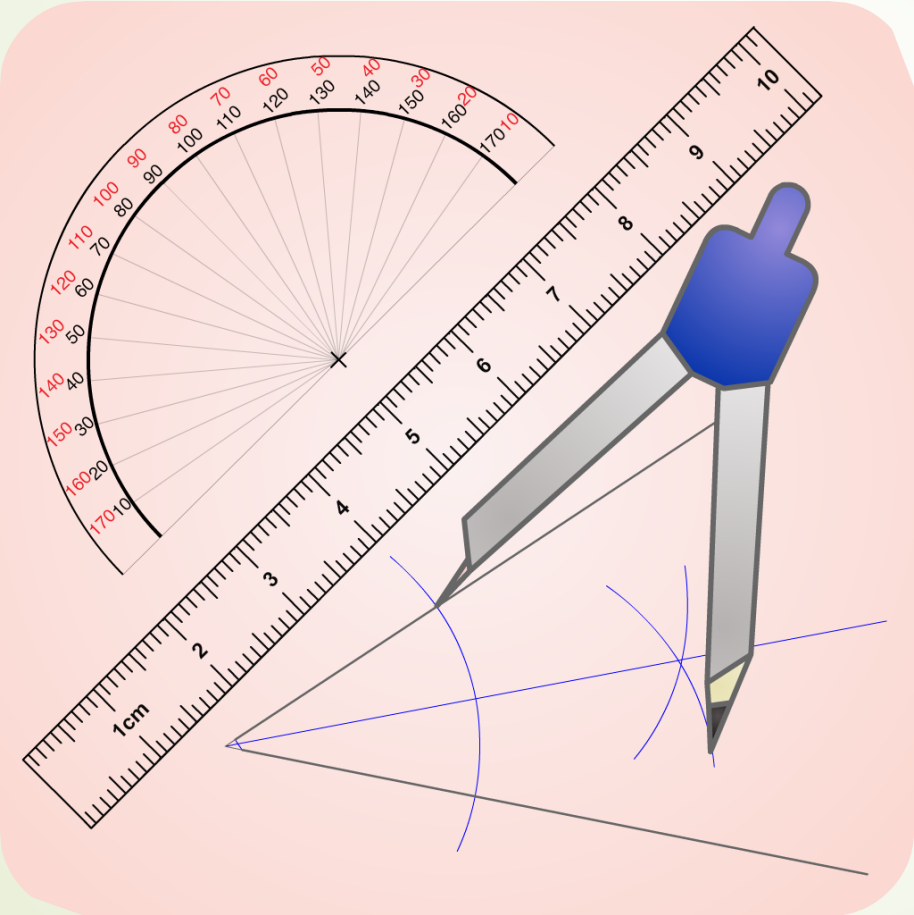
Sketchpad Graphics Language

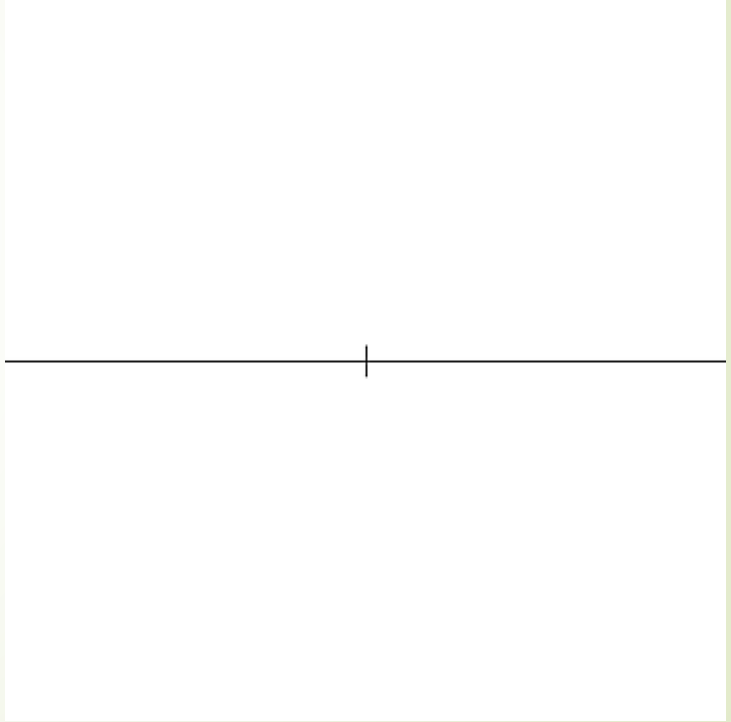
Yan Peng, Yichen Liu, Zhongyu Wang

Overview

- Compass-and-straightedge construction
- Dependence Relationship
- Subfunction
- Symbol Table
- Statically scoped, Byval and Byref.

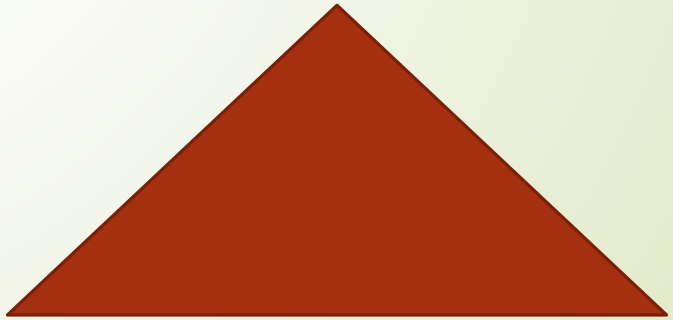
Simple, but Strong!





Geometry Statements

- Geometry Statements are usually not about a certain graph, but about a set of graphs in some certain constraint.
- E.g. the three perpendicular bisector of the three edges of a triangle meet at one point.
- Our language gives a easy way to check different instances in a certain constraint. :Move some part of the graph, the other parts will reshape.



Types and Operators

- integer, float, string, bool
- Point, Line, Circle.
- Basic arithmetic and logic operators.
- Basic library functions

Basic Syntax:

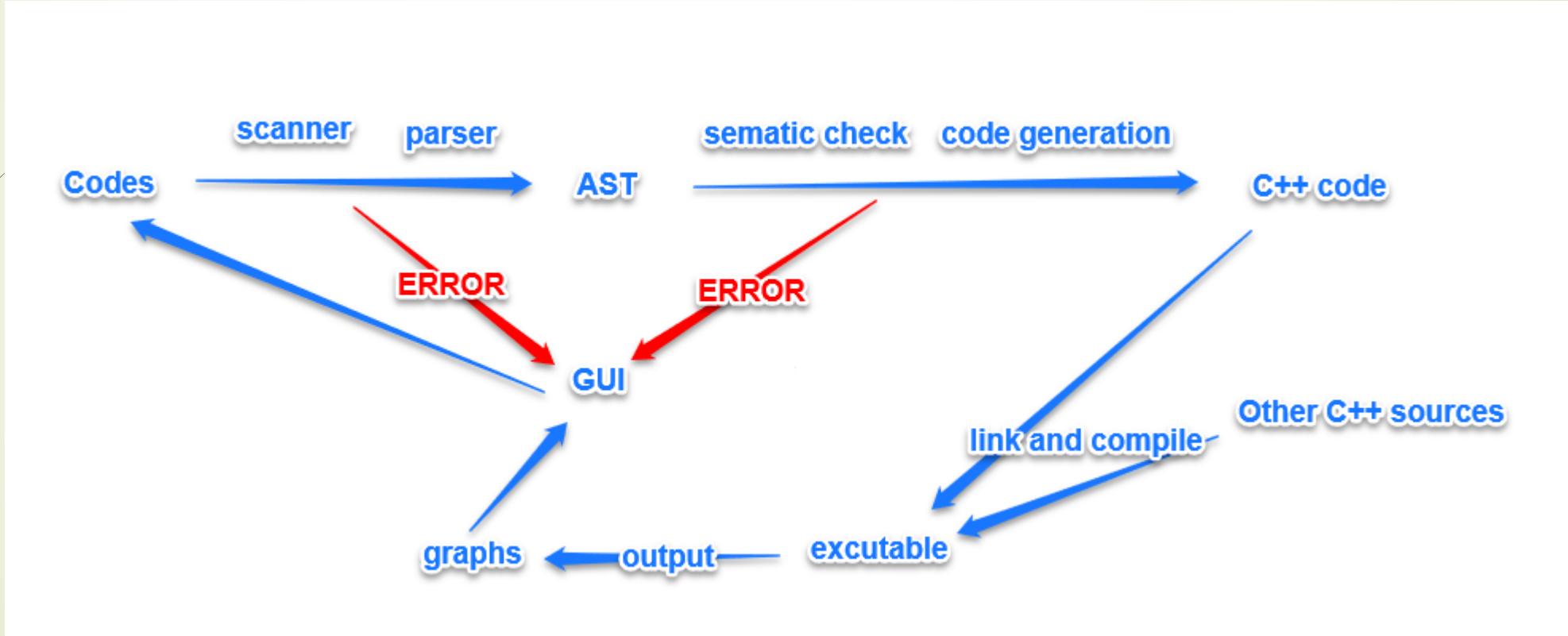
- A combination of C++ and VB

```
integer factorial(integer n)
    integer fib(integer n,integer prod)
        if n==1 then
            return 1;
        else
            return fib(n-1,N*prod);
        end
    end
    return fib(n,1.5);
end
```

```
Line PerpBisect(Point A, Point B)
    Line l1=library.LineST(A,B,2);
    Circle c1=library.DrawCircle(A,l1);
    Circle c2=library.DrawCircle(B,l1);
    Point C=library.intersect(c1,c2,true);
    Point D=library.intersect(c1,c2,false);
    Line l3=library.LineST(C,D,2);
    l1.setvisible(false);
    c1.setvisible(false);
    c2.setvisible(false);
    C.setvisible(false);
    D.setvisible(false);
    return l3;
end
```

```
integer fib (integer n)
    if n<3 then
        return 1;
    else
        integer a1;
        integer a2;
        integer a3;
        integer i;
        a1=1;
        a2=1;
        for i=3;i<n;i=i+1 do
            a3=a1+a2;
            a1=a2;a2=a3;
        end
        return a3;
    end
end
```

Structure of the program



Some Features

- sub functions
- Anywhere variable declaration
- Statically scoped
- By value and by reference.

Code generation

- Symbol Table: Managing Scopes
- Sketch Pad: managing dependency relationship
- Function translation: put them ahead, name them according to their appearance order.
- Code generator need to keep track of the environment:

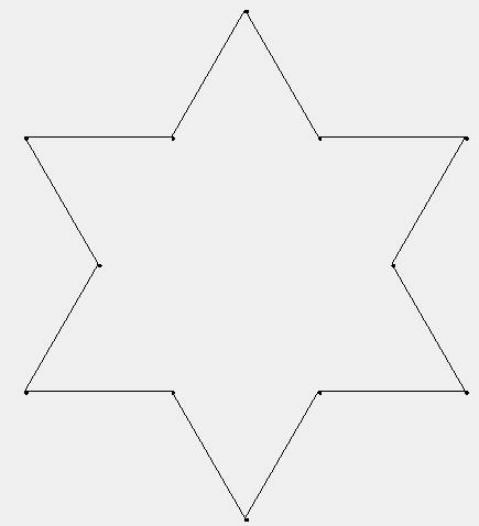
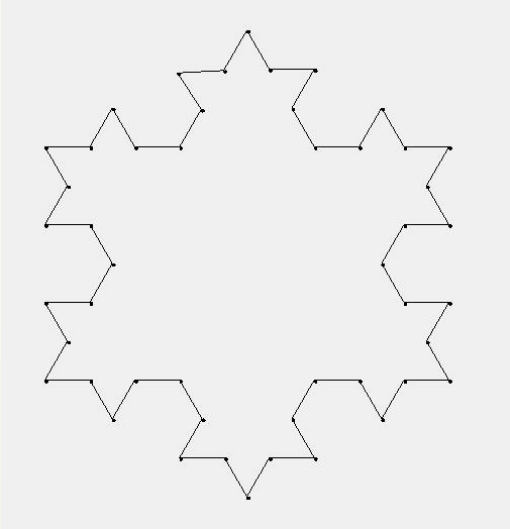
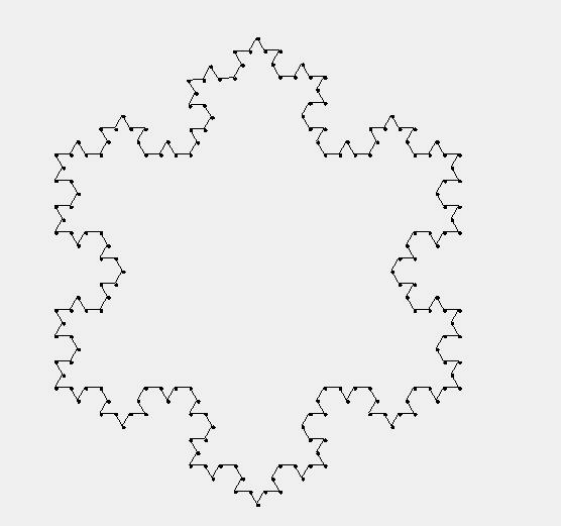
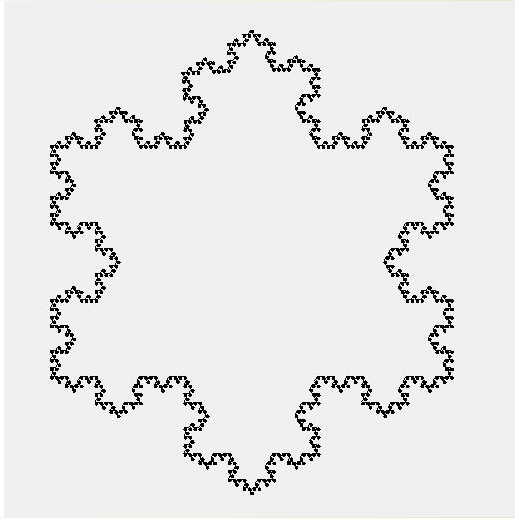
Code generation

```
int  function1(int n){
int _return_value;Symbol_Table_Node* parent=_CurSymbolTable;
Symbol_Table_Node* _CurSymbolTable=new Symbol_Table_Node(parent);
_CurSymbolTable->add_var(n,"n");
_CurSymbolTable->add_function((func_pointer) &function1,"fib");

_return_value=((int (*)(int,int))_CurSymbolTable->get_function("fib"))(*((int*)_CurSymbolTable->get_var("n")),1.5);
 goto endline;
endline:
delete _CurSymbolTable;
return _return_value;}
```

```
    GeoObj::~GeoObj(){
        if (this->parent[0]!=NULL){
            this->parent[0]->children->del(this);
        }
        if (this->parent[1]!=NULL){
            this->parent[1]->children->del(this);
        }
        delete this->children;
        sketch->eleList->del(this);
    }
```

Demo: snowflake fractal

```
integer snowflake(integer depth)
    integer iter(Point A,Point B,Point Center,integer depth)
        if depth==0 then
            library.LineST(A,B,0);
            return 1;
        end
        Point C=onethird(A,B,1);
        Point D=onethird(A,B,2);
        Line l1=library.LineST(A,C,0);
        l1.setvisible(false);
        Line l2=library.LineST(D,B,0);
        l2.setvisible(false);
        Circle c1=library.DrawCircle(C,l1);
        c1.setvisible(false);
        Circle c2=library.DrawCircle(D,l2);
        c2.setvisible(false);
        Point E=library.intersect(c1,c2,true);
        Point F=library.intersect(c1,c2,false);
        if Dist(Center,E)>Dist(Center,F) then
            F.setvisible(false);
        else
            E.setvisible(false);
            E=F;
        end
        Point O1=library.PointXY((C.getX()+D.getX()+E.getX())/3.0,(C.getY()+D.getY()+E.getY())/3.0);
        O1.setvisible(false);
        iter(A,C,Center,depth-1);
        iter(C,E,O1,depth-1);
        iter(E,D,O1,depth-1);
        iter(D,B,Center,depth-1);
        return 1;
    end
    Point A=library.PointXY(200.0,200.0);
    Point B=library.PointXY(600.0,200.0);
    Point C=library.PointXY(400,200+200*1.732050808);
    Point O=library.PointXY(400,200+200*1.732050808/3.0);
    O.setvisible(false);
    iter(A,B,O,depth);
    iter(B,C,O,depth);
    iter(C,A,O,depth);
    return 1;
end
snowflake(1);
library.paint();
```

Lessons Learned

- Tests are never enough
- Keep a good structure so that it can be easily modified when you have new ideas.
- Communication is important in group work

Thanks!