Sketch Graphic Language
Final Report

Yan Peng yp2321, Yichen Liu yl 2904,Zhongyu Wang zw2259

Contents

1. Introduction

1.1 Background

Compass-and-straightedge construction is the core part of mathematic in ancient Greece. With only several basic operations, compass-and-straightedge construction can build complex geometrical objects.

Geometrical statements can be presented on X-Y coordinates. One geometrical statements can correspond to a set of different graphs. For example: All three angle bisectors of the three angles in a triangle meet at one point. In this statement, the set of all kinds of triangles must be taken into consideration. To check whether a statement is true, people may need to produce a set of different graphs with the same constraint.

In order to produce many different graphs for one statement easily, we present a graph by "constraints" corresponding to operations in compass-and-straightedge construction. For example: "connect A,B" means draw a line from A to B. Here the line is not presented by a certain intercept and slope. When the position of A or B changes, the line's intercept and slope will also change so that it still go through A and B.

1.2 Language Features

Because there might be many repeated operations to finish a certain construction (like "Draw the centroid of a triangle"), a strong data encapsulation ability is required in the language.

Our language allows sub-function definition, which means we can define a function inside a function. What's more, our language allows anywhere variable declaration like C++ rather than C, for convenience.

2. Language Tutorial

Sketchpad Graphics Language (SGL) is a language for user to construct complex geomantic graphic for mathematical analysis. Student can use this language to solve the problems that are related to points, circles, and lines. Before using this tool to solve problems, user should learn several things to get familiar with this language.

2.1 Getting started with the SGL Compiler

Sketchpad Graphics Language (SGL) has its own interface to run the code, and the interface was built under Visual Studio C#. Therefore, the user should install Microsoft Visual Studio first, then the only thing the user need to do is to download the whole project, named "cdocnotepad", and open it as an C# project in the Visual Studio. Then user can start to use the SGL successfully.

2.2 Example

We are going to show an easy example step by step to make the user get more familiar with SGL.

(1) When user wants to start coding, the first thing we need to do is to identify the elements by using the identifier with only letters and digits. Be careful that the identifiers are case sensitive. The following code does the work that initialize two integers, i and j, and then use a "for" loop to assign j with the value of 0+1+2+3+4+5+6+7+8+9+10+11+12+13+14.

```
integer i;
integer j=0;
for i=0; i<15; i=i+1 do
        j=j+i;
end
```

(2) The next several lines of code are to show a user defined functions in SGL. The function name is "factorial" and the function parameter is an integer "n". We could define another function in the same scope that is the "fib" function. The

“fib” function needs two parameters, one is an integer “n” and the other is an integer “prod”. The “fib” function uses recursion and does the work of calculating the value of factorial of n and then the multiplication of this factorial and “prod”. Then the “fib” function returns the value of the multiplication. So in the “factorial” function, we set the value for the “prod” parameter to 1 and then call the function “fib”.

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

        end

    end
    return fib(n,1);
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=1;i<n;i=i+1 do
            a3=a1+a2;
            a2
        end
end
```

(3) The next several lines of code do the work that use the three basic types that defined by the SGL (Line, Circle, Point) to find the intersection points between two circles and connect the intersection points by a line. Firstly, we use the library function "LineST" to create a line from "A" point to "B" point. Secondly, we could create two circles that both use line "l1" as radius. The midpoint of "c1" is "A" and the midpoint of "c2" is "B". Then we create points "C" and "D" as the intersected points of circles "c1" and "c2". Then, we create the line "l3" as the line determined by points "C" and "D". Finally, then line "l3" is what we need.

```
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
```

(4) The next several lines of code are still easy to understand. The user-defined function "MidPoint" is to find the midpoint of two points, "A" and "B" which are the parameters of this function. It firstly finds the perpendicular line, "l1". Then "l2" is the line connected by A and B. Finally, it finds the point intersected by "l1" and "l2".

```
Point MidPoint(Point A, Point B)
    Line l1=PerpBisect(A,B);
    Line l2=library.LineST(A,B,2);
    Point midpoint=library.intersect(l1,l2,true);
    return midpoint;
end
```

(5) The next few lines are user-defined function to find the parallel line of “l” through point “A” and then return it. Line “l3” is the line through points “A” and “B”. Circle “c1” is the circle with radius “A” and diameter “BC”. Circle “c2” is the circle with radius “B” and diameter “AC”. Point “D” and “E” are the intersection points of circle “c1” and “c2”, and one of them is parallel with “BC”. Finally, the “if” condition is to find which point is not matched with point “A” and draw the line “l4” through points A and this point. The line “l4” is what we want and should be returned.

```
Line parallel(Point A, Line l)
    Line l2;
    if l.getLineType()==2 then
        l2=l;
    else
        l2=library.LineST(l.getStart(),l.getEnd(),2);
    end
    Point B=l2.getStart();
    Point C=l2.getEnd();
    Line l3=library.LineST(A,C,2);
    Circle c1=library.DrawCircle(A,l2);
    Circle c2=library.DrawCircle(B,l3);
    Point D=library.intersect(c1,c2,True);
    Point E=library.intersect(c1,c2,False);
    Line l4;

    if (D.getX()-A.getX())*(C.getY()-B.getY())-(C.getX()-B.getX())*(D.getY()-
A.getY())<0.01 then
        l4=library.LineST(A,D);
    else
        l4=library.LineST(A,e);
    end
    return l4;
end
```

(6)The next several lines are used for the user-defined function that return the one-third point in the line defined by the points “A” and “B”.

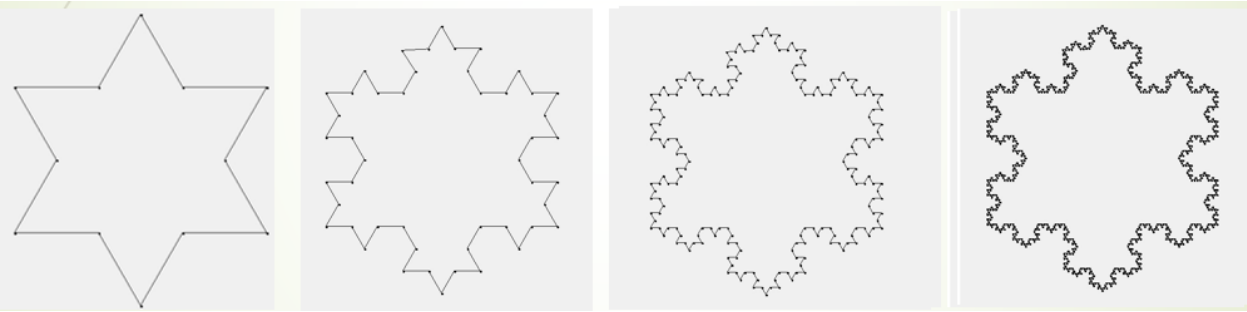
```
Point onethird(Point A, Point B,integer prop)
    Line l1=library.LineST(A,B,2);
    Circle c1=library.DrawCircle(A,l1);
    Circle c2=library.DrawCircle(B,l1);
    Point C=library.intersect(c1,c2,true);
    Line l2=library.LineST(A,C,2);
    Circle c3=library.DrawCircle(C,l2);
    Point D=library.intersect(c3,l2,true);
    Point E=library.intersect(c3,l2,false);
    Point F;
    if abs(D.getX()-A.getX())<0.01 && abs(D.getY-A.getY)<0.01 then
        F=E;
    else
        F=D;
    end

    Line l3=library.LineST(F,C);
    Circle c4=library.DrawCircle(F,l3);
    Point G=library.intersect(c4,l3,true);
    Point H=library.intersect(c4,l3,false);
    Point I;

    if abs(G.getX()-C.getX())<0.01 && abs(G.getY-C.getY)<0.01 then
        I=H;
    else
        I=G;
    end

    Line l4=library.LineST(B,I,2);
    Line l5;
    Point J;
```

```
        if prop==1 then
                l5==parallel(C,l4);
                J=library.intersect(l1,l5);
        else
                l5==parallel(F,l4);
                J=library.intersect(l1,l5);
        end
        return J;
end
```

sample program

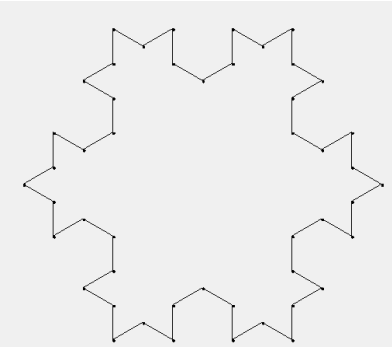
Sample Program: Snowflake Fractal

The source code is contained in the project file.



In the program, we used a function called snowflake, inside whom there is a sub-function called iter that recursively split each edge into snow-flake pieces.

After that, we changes the start point of the fractal and other parts moved automatically.



3. Language Reference Manual

3.1. Introduction

This manual provides reference information for using the SGL (Sketchpad Graphics Language). Sketchpad Graphics Language is a language designed for users to construct complex geometric graphics for mathematical analysis. This manual describes the lexical conventions, basic types scoping rules, built-in functions and grammar of SGL.

3.2 Lexical conventions

There are six kinds of tokens: identifiers, keywords, constants, strings, expression operators, and other separators. Space, tabs and newlines are symbols of separating tokens. Every token should match the longest string that could comprise a token in

the input buffer. For example, if there following two characters in the input buffer are “~” and “=”, we will get the token from “~” rather than from “=”.

3.2.1 Comments

is used for comments. Also, only one-line comments is legal in our language.

3.2.2 Identifiers (Names)

An identifier is a sequence of and only of letters and digits. An identifier must begin with a letter. ALG is a case sensitive language, so uppercase and lowercase are different for the compiler.

3.2.3 Keywords

The following identifiers are reserved for use as keywords, and may not be used otherwise:

integer	do	Line
string	for	Circle
float	while	Point
bool	break	library
if	continue	true
then	function	false
else	return	
elif	pass	
end		

3.2.4 Constants

There are several kinds of constants, as follows:

3.2.4.1 Integer constants

An integer constant is a sequence of digits from 0 to 9. Only decimal integer is supported in our language. It takes 64 bit and is signed.
Corresponding regular expression: ['0'-'9']+

3.2.4.2 Floating constants

A floating constant consists of an integer part, a decimal point and a fraction part. The integer part and fraction part either consists of a sequence of digits and must have at least one digit. It is a 64-bit floating point number which is similar with double in C. Only decimal representations are allowed.

Corresponding regular expression: ['0'-'9']*'.'['0'-'9']+

3.2.4.3 Bool constants

Bool type only includes two possible values: true and false.

Corresponding regular expression: “true”|”false”

3.2.4.4 String constants

A String is a sequence of characters surrounded by double quotes “ " ”. String are mainly used as labels in our program so a single character in the string can’t be changed or visited. The only operation of String is concatenation using “+”operator.

3.2.5 Operators

+	Addition
-	Subtraction(both a bi-operator and a uni-operator)
*	Multiplication
/	Division
~	Not
&&	And
||	Or
==	Equality
>=	Greater or equal
<=	Less or equal
>	Greater than
<	Less than
!=	Not equal to
=	Assignment

3.2.6 Ponctuations

The following symbols are used to organize code and to specify different organizations of objects:

Symbol	Definition
; ,	Marks the end of a statement Function declaration and calling; and specifies operation priority

3.3. Object Types

There are two broad classes of objects supported in SGL.

3.3.1 Basic types

There are three basic types defined by the Sketchpad Graphics Language. Type identifiers always begin with an upper-case letter followed by a sequence of one or more legal identifier characters. The built-in types include:

- Point
- Line
- Circle

These objects have properties and methods to modify their values.

3.3.1.0 Universal Properties for geometry objects

integer getcolor()	get color shown on the graph
integer setcolor(integer color)	set color shown on the graph
bool isvisible()	whether the point can be seen on current graph
bool setvisible(bool visible)	set whether the point can be seen on current graph
string getname()	get name shown on the graph
string setname(string name)	set name shown on the graph
void unlock()	free it from its parents

3.3.1.1 Properties for Point

Type and name	definition

float getX()	get x-coordinate of the point
float getY()	get y-coordinate of the point
void setXY(float X,float Y)	

3.3.1.2 Properties for Line

float getX1()	get x-coordinate of the first point
float getY1()	get y-coordinate of the second point
float getX2()	get x-coordinate of the first point
float getY2()	get y-coordinate of the second point
integer gettype()	the type of the line: segment, ray or line

3.3.1.3 Properties for Circle

float getX()	get x-coordinate of the center
float getY()	get y-coordinate of the center
float getradius()	get the length of radius
float setradius()	set the length of radius
void setXY(float X,float Y)	set the position of the center

3.3.2 Atom types

There are four Legal atom-types corresponding to 4 basic constants: Integer, Float, Bool and String

3.4. Expressions and Operators

In this section we describe the built-in operators for SKL and define what constitutes an expression in our language. Operators are listed in order of precedence. All operators associate left to right, except for assignment, which associates right to left.

3.4.1 Primary expressions

3.4.1.1 identifier

See section 2.2.

3.4.1.2 constant

See section 2.4.

3.4.1.3 (expression)

A parenthesized expression is a primary expression whose type and value are identical to those of the unadorned expression.

3.4.2 Unary operators

Expressions with unary operators group right-to-left.

3.4.2.1 ~ expression

The result of the logical negation operator ~ is a bool. It performs negation on the expression. This operator is applicable only to integers.

3.4.3 Multiplicative operators

The multiplicative operators *, and /, all group let-to-right.

3.4.3.1 expression * expression

The binary * operator indicates multiplication. It is valid between two integers, two floats, an integer and a float, or a Point and an integer or float. If one is an integer and the other is a float, the result is a float. If one is a Point, its x and y values are each multiplied by the other expression.

3.4.3.2 expression / expression

The binary / operator indicates division. The same type considerations as for multiplication apply. Division by zero is not allowed.

3.4.4 Additive operators

The additive operators + and - group left-to-right.

3.4.4.1 expression + expression

The result is the sum of the expressions. Addition is valid between two integers, two floats, an integer and a float, or two Points. In the case of addition of an integer and a float, the result is a float.

3.4.4.2 expression – expression

The result is the difference of the operands. Subtraction is defined identically to addition.

3.4.5 Relational operators

The relational operators group left-to-right.

3.4.5.1 expression < expression

3.4.5.2 expression > expression

3.4.5.3 expression <= expression

3.4.5.4 expression >= expression

The operators < (less than), > (greater than), <= (less than or equal to), >= (greater than or equal to) all yield a bool, either true or false. The relational operators are valid for comparison between two integers, two doubles, or a float and an integer.

3.4.6 Equality operators

3.4.6.1 expression == expression

Equal to.

3.4.6.2 expression ~= expression

Not equal to.

The equality operators are valid for comparison between two integers, two floats, a float and an integer, or two Points. The result is a bool. When comparing the

equality of two points, the result is true is the x and y values of the two points are identical.

3.4.7 Logical operators

3.4.7.1 expression && expression

Logical AND between two bool expressions.

3.4.7.2 expression || expression

Logical OR between two bool expressions.

3.4.8 Assignment operators

3.4.8.1 lvalue = expression

The value of the expression replaces the value of the object that the lvalue refers to. Types must match.

3.4.9 Conversions

A number of operators may, depending on their operands, cause conversion of the value o an operand from one type to another. Multiplication of an integer and a float will be a float. The rule also applies to division, addition, and subtraction.

3.5. Declarations

3.5.1 Variable declaration

Each variable must be declared before used.

Integer:

```
integer a = 10;
```

Float:

```
float a = 10. ;
```

```
float a = 1.0;
```

Bool:

```
bool a = true;
```

String:

```
string s = " Hello! ";
```

Point:

```
Point A;
```

Line:

```
Line L1;
```

Circle:

```
Circle O;
```

3.5.2 Function declaration

The SKL language supports C++-like user-defined functions.

Functions must be declared and implemented meanwhile, before they are called by users.

return type function name (parameter type parameter name, …) # function declaration

function implementation

end

Example:

```
integer fib(integer n)
    return fib(n-1)+fib(n-2);
end
```

3.6. Statements

Except as indicated, statements are executed in sequence.

3.6.1 Expression statement

Most statements are expression statements, which have the form

expression ;

Usually expression statements are assignments or function calls.

3.6.2 Conditional statement

The two forms of the conditional statement are

if *expression* then *statement* elif *expression then statement* …… else *statement end*

if *expression* then *statement* else *statement end*

if *expression* then *statement end*

The program check all the expressions from first to last and find the first true expression, execute corresponding sub-statements. If none is satisfied, the program execute the else statements.

3.6.3 While statement

The while statement has the form

while *expression* do *statement end*

The sub-statement is executed repeatedly so long as the value of the expression remains true.

The test takes place before each execution of the statement.

3.6.4 For statement

The for statement has the form

for *expression1* ; *expression2*; *expression3* do *statement end*

Thus the first expression specifies initialization for the loop; the second specifies a test, made before each iteration, such that the loop is exited when the expression becomes true; the third expression typically specifies an incrementation which is performed after each iteration. Any or all of the expressions may be dropped.

3.6.5 Break statement

The statement

break ;

causes termination of the smallest enclosing while, or for statement; control passes to the statement following the terminated statement.

3.6.6 Continue statement

The statement

continue ;

causes control to pass to the loop continuation portion of the smallest enclosing while, or for statement; that is to the end of the loop.

3.6.7 Return statement

A function returns to its caller by means of the return statement, which has one of the forms

return ;

return (expression) ;

3.7. System Functions

Keyword library is preserved and is used for calling built-in functions.
The library functions are called using library.*functionanme*
The function includes:

Line LineST(Point P1,Point P2,integer line_type);
Circle DrawCircle(Point center, Line radius);
Circle DrawCircle(Point center, double radius=NONEXIST);
Point intersect(GeoObj GeoObj1,GeoObj GeoObj2,bool isnear);
void paint();
Point PointXYfloat X,float Y);
Line LineXY(float X1,float Y1,float X2,float Y2,integer line_type);
Circle CircleXYr(float X,float Y,float r);

3.8. Scoping Rules

A scope in any programming is a region of the program where a defined variable can have its existence and beyond that variable can not be accessed. There are three places where variables can be declared in SGL:

- Inside a function or a block which is called local variables/functions,

  The block contains: the sub-statements of conditional statements, for loops and while loops

- Outside of all functions which is called global variables/functions.

- In the definition of function parameters which is called formal parameters.

The program contains sub-functions and the scoping rule is thus more complicated.

If an identifier appears, the language first try to find it in its local scope, if it can't be found then the language turns to its parent-scope to find the identifier. What is worth noticing is that the parent of any function is just the global environment.

```
integer a #visible to c

integer b()

    integer a;#not visible to c

    integer c()

        blabla…..


    end

end
```

4. Project Plan

Our team was formed in late September. After that we meet at least three times a week, that is Monday, Wednesday and Saturday. The topics for each meeting varies as we made progress in the projects, but the procedure is standard. Every time in the meeting, we shall first check whether we have accomplished all the tasks that were assigned in the previous meetings and discuss how to solve the obstacles preventing us from moving forward. Then, we shall assign new tasks to each member of us according to the project timeline set before. Last, we will talk about the content of the PLT courses to make sure everyone is in pace with the lectures and has a good understanding of we are doing in the project.

We adopt Github to help us control the code version. Every member of the group contributes to the PLT repository once the code is accomplished and we do updates frequently to fix bugs and make optimizations. Since this is the first time we use adopt a SVN, we find it not that convenient to help share and control the code, because we usually encounter the problem of not being able to checkout the latest version and have to make clones almost every time. But still we forced us to do use Github because it has become an essential skill in our future study and work. In the end of the project, we can work well with Github and find it truly brings much convenience to our project. It's cool!

Our development strategy is an iterative process. To be specific, we will test our codes along the whole process in order to ensure we will not bring any independent bugs to the next step. The strategy works well in our project because this will narrow down the scope of our bug-checking. Though we have to modify previous files as we approach the finalized version, this strategy truly helps save a lot of time to concentrate on the newly-born bugs.

Besides conducting unit test for each of the file, our testing strategy mainly focuses on the semantic check phase and the final testing phase. Semantic check is trivial and we have to test whether all the legal input types will be admitted and illegal ones denied. It takes a lot of time and we need to be very careful about it. During the final testing phase we build two testing sets, one is for legal input and the other illegal input. More details about how we conduct these tests will be provided in the following corresponding chapter.

4.1 Planning

After we completed the language reference manual, our team met every Sunday to discuss the project, check our progress. Also, we decided what we should do for the upcoming week. This process worked perfectly to help the group members worked together and solved any issues that happened during the design. Also, we used chat software to contact each other anytime we need. During the same time, we shared each member's work on Google doc to make sure every member can see it and use it. We found that this was the most efficiently way for our group.

4.2 Specification

After we completed the language reference manual, we immediately worked on the compiler part. The specification of the compiler changed as we realized the difficulty of implementing the features during a short time. The changes are noted in part 3 of this document.

4.3 Development

First of all, we wanted to have an iterative approach where we added a feature then tested it to see it is worked or not, then we added another feature and do the same process as before. Then we found that the scanner and parser were created before the code generation part. Then the code generation module was fully functional before the semantic analysis was started. The "gen.bat" is a batch file which create scanner, parser, code generation and finally it generates result.cpp file. Since the development environment interface was written by C#, so we must include the C++ bin here to use the command "cl.exe" to change the ".cpp" files to ".obj" files and then use the command "ml.exe" to change the object files to executable file which could output a ".txt" file. We then read the text file and draw the corresponding image.

4.4 Programming Style

Even though the programming environments vary among the group due to different operating systems, still we attempt to maintain the programming style similar to that of MicroC, which is introduced in the class. Also we wrote comments for almost every function in the semantic check part, because the types to be checked are trivial and numerous, as for other parts, we keep the format neat and keep fewer comments for these files because they are relatively easy to understand.

4.5 Project Timeline

Date	Goal
September 26, 2013	Project proposal, core language features defined
October 5, 2013	Began work on the language reference manual
October 20, 2013	Code convention, group member roles and work

	area defined
October 28, 2013	Language reference manual, grammar, scanner completed
November 17, 2013	Parser completed
December 12, 2013	Semantic analysis and environment interface completed， code generation completed
December 20, 2013	Submitted final project, project files, and presentation

4.6 Responsibility

Below is the responsibility for each group member:

Group Member	Responsibility
YiChen Liu	Semantic check
Yan Peng	Code generation, develop environment interface
ZhongYu Wang	Scanner, parser, code generation

4.7 Environment

4.7.1 Language

We use O'Caml, C++, Csharp and our own language SKL in our project to compose the files. Specifically, we use O'Caml to write the scanner, parser, ast, semantic check, and code generation parts; we use C++ to write the implementation of the built-in geometric functions; we use Csharp to make the executable programs and SKL to write our testing cases.

4.7.2 Environment

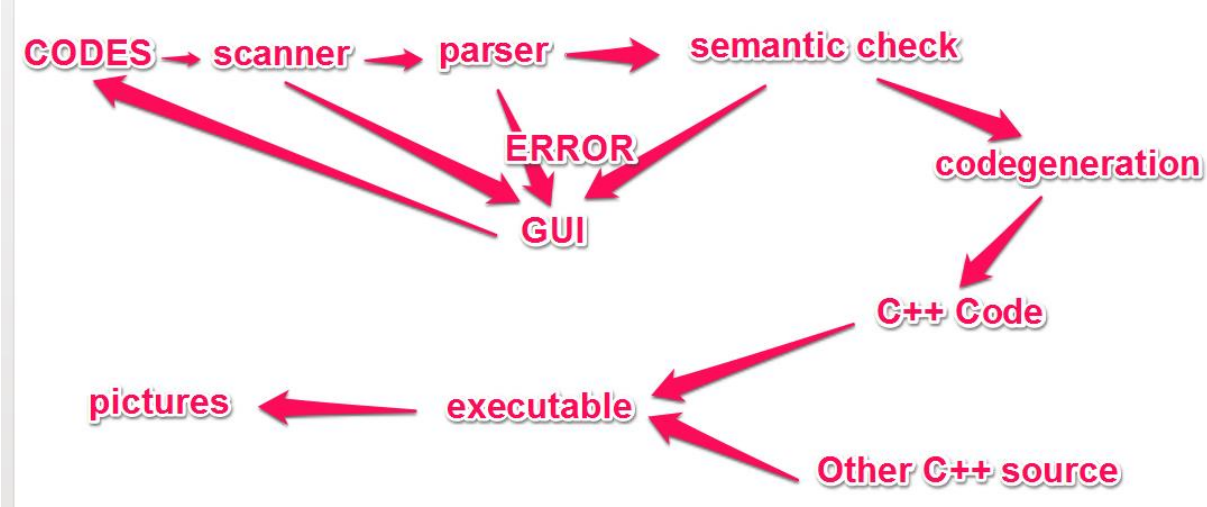
Our computers run in different operating systems: Mac OS X, Ubuntu 12.04 and Windows Vista, thus we adopt different developing environments for the tasks, they are XCode, gcc, Eclipse, and gcc47 for Mac. Also, we use ocamllex for lexical analysis and ocamlyacc for the parsing. Moreover, to realize all the functions, we also need to configure OpenCV for the developing environment and get "pkgconfig" installed in Mac. (For how to configure OpenCV in Mac, Please consult http://opencv.willowgarage.com/wiki/Mac_OS_X_OpenCV_Port.)

The compiler is primarily written in O'Caml and we write header file in C++ to implement the back-end built-in functions. All of these codes is written in editors like XCode, Vim and eclipse and ultimately compiled in gcc47 for Mac.

Also we adopt other tools in the project to make us get organized. To be specific, we use github to share and control different version of our codes and get everyone updated with the latest version. We use Google Doc to share documents and reading materials among group members and also established a Tencent Discuss Group talk about project problems remotely. All these tools make each of us informed about our progress and help our project work efficiently.

5. Architectural Design

Here is a diagram to illustrate the main components and steps of SKL projects. Detailed explanation will be in other subsections of this section.



5.1 Scanner

The scanner part will convert the raw SKL language into token stream, which will be used in the parser to compose the AST. It will also recognize the comments in the program so that the compiler will ignore them.

5.2 Parser and AST

In this part, we define our syntax and build up the AST. The input of token stream will finally be converted into two lists: a global variable declaration list, and a function declaration list. In the global variable declaration list, each global variable will be stored in a tuple whose form is (vari_type * string), so that we can get access to this variable's type and its name. In the function declaration list, we store each function in a type who has the following properties: function name, function return type, function parameter list, function local variable list, and function body.

With the above two lists, we simulate an AST and in the following Semantic check part, we can recursively check the validity of AST structure.

5.3 Sematic check

The semantic analyzer includes the important features like:

1. check the duplication of variables in global/local variables
2. duplication of function names

3. validity of expression
4. validity of statements
5. validity of function declaration
6. validity of operators

We start from “errors” and see what the semantic analyzer should do to stop those errors before the AST is passed to code generator. Typical errors are: Duplicated function name/ variable name/ parameter name, wrong number of parameters when using a function, non-defined function, wrong expression type in assign or function call, wrong statement form, lack of return statement in a function, no main function, etc.

If there is any error in the SKL source code that has passed the parser section, it would be hijacked by the semantic analyzer and exception will be threw to inform the user which kind of error has been detected.

5.4 Code Generation

Code generation translate AST into C++ codes. All parts, except function call/definition and variable definition/assignment/value can be easily translated to corresponding part in C++.

Because our scoping rule is quite different with C++, there is no point just translating them to C++ ways of definition/declaration etc.

In order to implement that, we build a symbol table and a sketchpad (a dependency directed acyclic graph). When a scope is built, a new node in symbol table is created and linked to its mother scope. And when finished, it is deleted. Variable/function definition corresponds to adding new definitions to current symbol table node and in function call/assignment/getting value of a variable, the program search in the symbol table to find the identifier and carry out corresponding operation.

Function definition is tricky. C++doesn’t allow function definition inside a function. What we do is to put them ahead in front of the main function in generated code and give them new names according to a integer maintained in the environment such as ‘function 1’, ‘function 2’ other than their original name in the

SKL language to avoid name conflict. Function call is maintained by our symbolist and the locality is ensured by the symbol table, not the C++ scoping rules.

To do this in code generation, the program walks through the AST and maintains the environment (locals, globals and function counts, etc.) so that what we need are available when needed

5.5 C++ implementation

There are 3 main files: stdafx.h, stdafx.cpp and the generated code; stdafx.h contains all declarations of needed variables, enums and classes. Their definition can be found in stdafx.cpp.

Two classes are used to maintain the variables in source language: symbol table and sketchpad.

6. Test Plan

We adopted unit test method throughout our project in each components (Scanner, Parser, Semantic Analyser, Code Generator, C++ Library, Compiler). After the accomplishment of each component, we also did integrated testing to check whether the whole system fulfills the initial requirement, and also we went into the detail code to do white box testing to pick up bugs.

For example, in the coding process of Semantic analyzer, each low-level function will have a detailed goal. After writing one function, there should always be a test for it to see whether it will achieve our goal. The author manually composed varieties of expressions from naive ones to complicated one which included sub-expressions inside. In this way, we can always be confident to use the old components when we try to develop new components.

After the integration of whole system, we write SKL language and give it to the compiler directly to see whether it passes the initial check and give the expected results. These test suites can be divided into three parts: Semantic Test, Basic Function Test, and Advanced Function Test.

6.1 Snowflake

The main test we want to show is to obtain a snowflake by using the SGL. The entire code and the output already show below:

```
float abs(float x)
        if x>0 then
                return x;
        else
                return -x;
        end
end
integer factorial(integer n)
        integer fac(integer n,integer prod)
                if n==1 then
                        return prod;
                else
                        return fac(n-1,n*prod);
                end
```

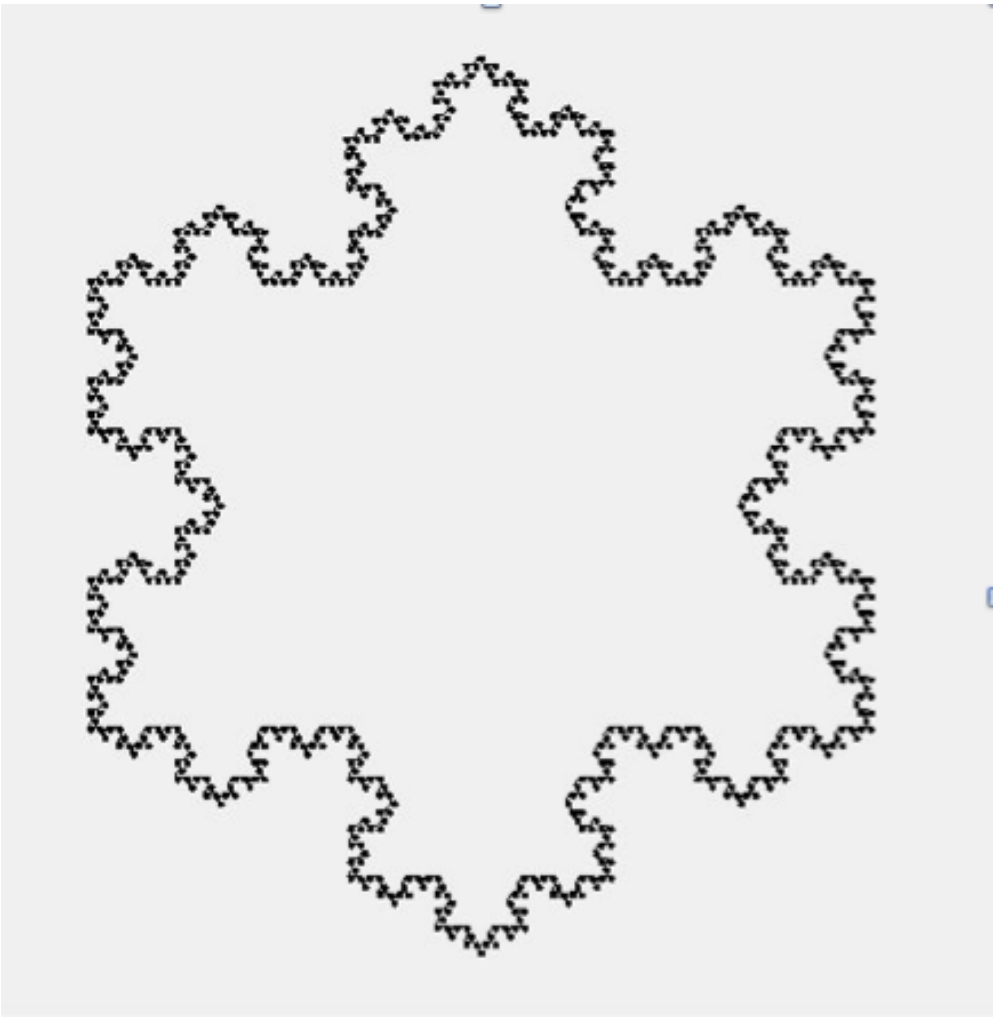
```
        end
        return fac(n,1);
end
integer fib (integer n)
        if n<3 then
                return 1;
        else
                integer a1;
                integer a2;
                integer a3;
                integer i;
                a1=1;
                a2=1;
                for i=3;i<n;i=i+1 do
                        a3=a1+a2;
                        a1=a2;
                        a2=a3;
                end
                return a3;
        end
end
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
Point MidPoint(Point A, Point B)
        Line l1=PerpBisect(A,B);
```

```
        Line l2=library.LineST(A,B,2);
        Point midpoint=library.intersect(l1,l2,true);
        return midpoint;
end
Line parallel(Point A, Line l)
        Line l2;
        if l.getLineType()==2 then
                l2=l;
        else
                l2=library.LineST(l.getStart(),l.getEnd(),2);
                l2.setvisible(false);
        end
        Point B=l2.getStart();
        Point C=l2.getEnd();
        Line l3=library.LineST(A,C,2);
        l3.setvisible(false);
        Circle c1=library.DrawCircle(A,l2);
        c1.setvisible(false);
        Circle c2=library.DrawCircle(B,l3);
        c2.setvisible(false);
        Point D=library.intersect(c1,c2,true);
        D.setvisible(false);
        Point E=library.intersect(c1,c2,false);
        E.setvisible(false);
        Line l4;
        if (D.getX()-A.getX())*(C.getY()-B.getY())-(C.getX()-B.getX())*(D.getY()-
A.getY())<0.1 && (D.getX()-A.getX())*(D.getY()-A.getY())>0.1 then

                l4=library.LineST(A,D,2);
        else
                l4=library.LineST(A,E,2);
        end
        return l4;
end
Point onethird(Point A, Point B,integer prop)
        Line l1=library.LineST(A,B,2);
        l1.setvisible(false);
```

```
        Circle c1=library.DrawCircle(A,l1);
        c1.setvisible(false);
        Circle c2=library.DrawCircle(B,l1);
        c2.setvisible(false);
        Point C=library.intersect(c1,c2,true);
        C.setvisible(false);
        Point GG=library.intersect(c1,c2,false);
        GG.setvisible(false);
        Line l2=library.LineST(A,C,2);
        l2.setvisible(false);
        Circle c3=library.DrawCircle(C,l2);
        c3.setvisible(false);
        Point D=library.intersect(c3,l2,true);
        D.setvisible(false);
        Point E=library.intersect(c3,l2,false);
        E.setvisible(false);
        Point F;
        if abs(D.getX()-A.getX())+abs(D.getY()-A.getY())<abs(E.getX()-
A.getX())+abs(E.getY()-A.getY()) then
                F=E;
        else
                F=D;
        end
        Line l3=library.LineST(C,F,2);
        l3.setvisible(false);
        Circle c4=library.DrawCircle(F,l2);
        c4.setvisible(false);
        Point G=library.intersect(c4,l2,true);
        G.setvisible(false);
        Point H=library.intersect(c4,l2,false);
        H.setvisible(false);
        Point I;
        if abs(G.getX()-C.getX())+abs(G.getY()-C.getY())<abs(H.getX()-
C.getX())+abs(H.getY()-C.getY()) then
                I=H;
        else
                I=G;
```

```
        end
        Line l4=library.LineST(B,I,2);
        l4.setvisible(false);
        Line l5;
        Point J;
        if prop==1 then
                l5=parallel(C,l4);
                J=library.intersect(l1,l5,true);
        else
                l5=parallel(F,l4);
                J=library.intersect(l1,l5,true);
        end
        l5.setvisible(false);
        return J;
end
float Dist(Point X1,Point X2)
        return (X1.getX()-X2.getX())*(X1.getX()-X2.getX())+(X1.getY()-
X2.getY())*(X1.getY()-X2.getY());
end
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(4);
  library.paint();
```



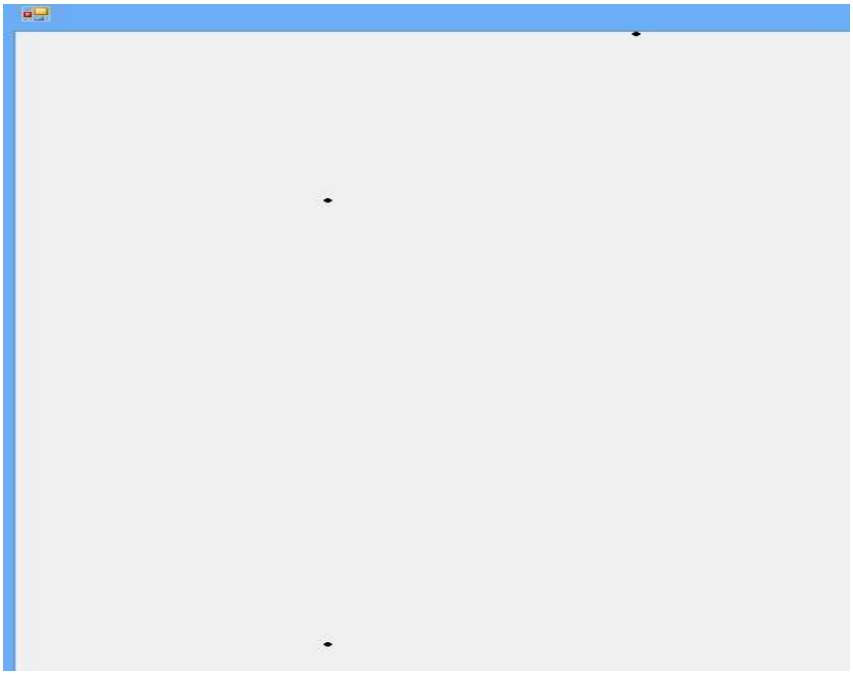
6.2 Points

This part shows how to plot points via SGL. In this example, we plot three points at different locations. User can plot anywhere they want by changing the coordinates. In our example, we plot point A at (150.0, 150.0), point B at (150.0, 550.0), and point c at (300.0, 0.0), and the figure shows the output clearly.

```
integer i;
integer j=0;
for i=0;i<15;i=i+1 do
        j=j+i;
end
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
Point MidPoint(Point A, Point B)
        Line l1=PerpBisect(A,B);
        Line l2=library.LineST(A,B,2);
        Point midpoint=library.intersect(l1,l2,true);
        return midpoint;
end
Line parallel(Point A, Line l)
        Line l2;
        if l.getLineType()==2 then
            l2=l;
        else
            l2=library.LineST(l.getStart(),l.getEnd(),2);
        end
        Point B=l2.getStart();
        Point C=l2.getEnd();
        return l;
```

```
end

Point A=library.PointXY(150.0,150.0);
Point B=library.PointXY(150.0,550.0);
Point C=library.PointXY(300.0,0.0);

  library.paint();
```



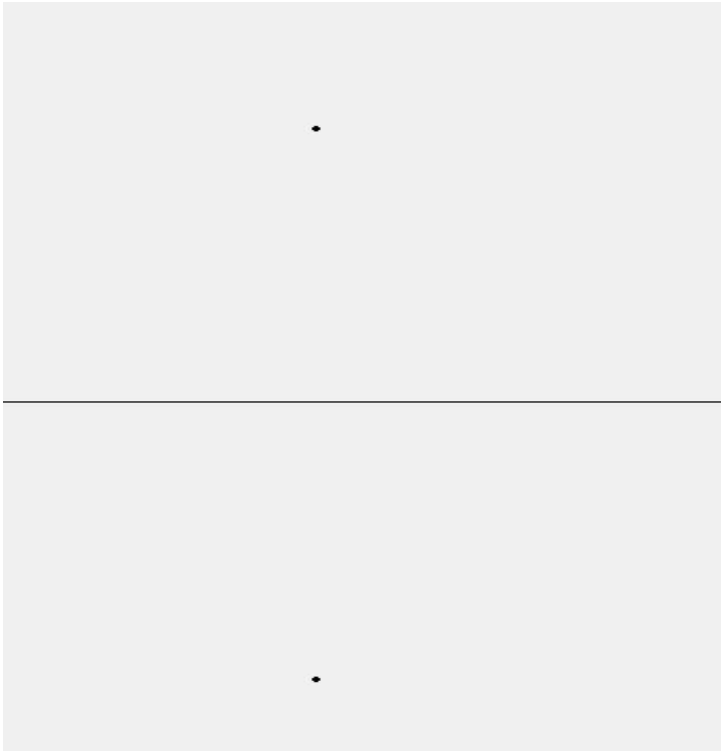
6.3 Perpendicular Bisector

To find the middle point between two points, we have to add a code to make it happens. The code is "Line L1=PerpBisect(A,B);". It gives the perpendicular bisector between point "A" and "B" as the figure shows below:

```
integer i;
integer j=0;
for i=0;i<15;i=i+1 do
        j=j+i;
end
integer factorial(integer n)
        integer fib(integer n,integer prod)
                if n==1 then
```

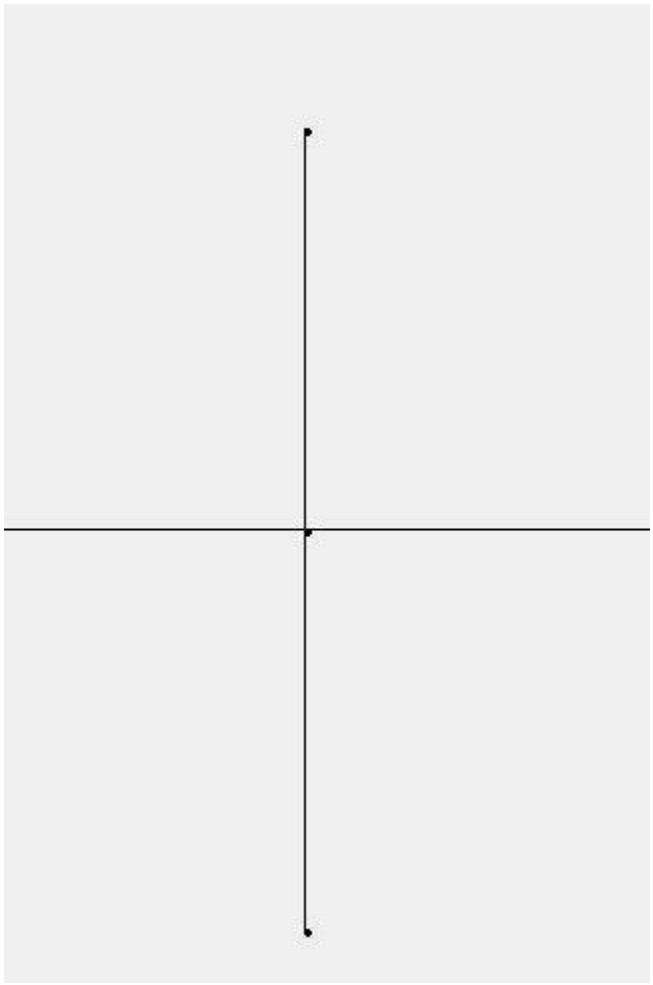
```
                return 1;
            else
                return fib(n-1,n*prod);
            end
        end
        return fib(n,1.5);
end
Line PerpBisect(Point A, Point B)
        Line l1=library.LineST(A,B,2);
        Circle c1=library.DrawCircle(A,l1);
        Circle c2=library.DrawCircle(B,l1);
        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
Point MidPoint(Point A, Point B)
        Line l1=PerpBisect(A,B);
        Line l2=library.LineST(A,B,2);
        Point midpoint=library.intersect(l1,l2,true);
        return midpoint;
end
Line parallel(Point A, Line l)
        Line l2;
        if l.getLineType()==2 then
            l2=l;
        else
            l2=library.LineST(l.getStart(),l.getEnd(),2);
        end
        Point B=l2.getStart();
        Point C=l2.getEnd();
        return l;
```

```
end

Point A=library.PointXY(150.0,150.0);
Point B=library.PointXY(150.0,550.0);
Point C=library.PointXY(300.0,0.0);
Line L1=PerpBisect(A,B);
library.paint();
```



6.4 Middle Point

We also can find the middle point between two points by adding the code “Point D=MidPoint(A,B);” where “A” and “B” are the points we used above. The figure of the output shows below:



7. Lessons Learned

7.1 Yichen Liu

It is such a pleasure to work with my teammates in this project. Building up a language compiler is a middle-sized project and required each teammate to understand almost all the components in this system. At first, I thought all I need to do is to understand how to code my own part (Semantic analyzer). However, I found that without the knowledge of scanner, parser, and the following C++ code generator, I could not move even one step forward. For example, when I was writing my own part of semantic check, I did several changes in the finished parser and AST, while it turns out that the change would require much modification in code generation also. Obviously, without a full understanding of every part of the compiler and by which method every component in it is related to each other, the team cannot cooperate the project successfully.

This project really helped me gain a deeper understanding of the general programming language implementation system. Coding with Ocaml for the Semantic Analyzer is convenient, even though sometimes bugs inside are not easy to find compared with C++ and Java. By viewing the code from last year programmers, I concluded several kinds of common errors in our language that Semantic analyzer should specially focused on.

Zhongyu did a wonderful job in finishing the scanner and parser early, and the robust code made it very comfortable for me to do some unit tests on my semantic analyzer. Also, Yan designed a perfect User Interface using Csharp, which saved us a lot of job to write makefile and enabled the test results straightforward. It was efficient to work with this version control system when we worked separately. Thanks a lot to Professor Stephen Edwards and all my teammates for this wonderful class.

7.2 Yan Peng

My job in this project is to generate the code and develop the environment interface. First I found that there are many ways to do the environment interface. But most of them are complicated. By doing some research, I finally decided to use C# to achieve the goal. Because this is the first time for me to use C#, the most challenged thing is to learn a new language in a very short time. I learned the

language while I coded. Even though it costs me more time to finish my job, I still feel happy and satisfied when I saw my code worked perfectly. For me, it is not only a project but also an opportunity to learn a new useful language.

For the code generation, it is so painful to think how to transform ocaml to c++. It has been some confusing time which is spent for discussing with my teammates and trying to figure out how to implement it. Moreover since the interface is written by C#, I must think how to change to c++ codes to the C# codes which could give the user with ouput image. At first time, I included the VC folder and use the “cl.exe” and “link.exe” commands, and try to produce the the executable file that gave the “result.txt” file which is used for draw the image. However, I found that I always missed some functions which caused errors. This really frustrated me, however, after browsed related Visual Studio tutorial file, I achieved the result by using after command, “devenv.com \RunExit”, which is automatically running the c++ project and then shutting down the IDE.

During this project, I also learned how to work with a team. People worked together, shared ideas, solved issues. I found that good team chemistry was really helpful to keep a project worked step by step and complete each part on time. Also, I learn that when we have a project, we really need to start it as soon as possible. We cannot just put everything on the last minute. People need to start it early to give more time to us to check the project and fix all issues and finally, we can have a perfect project as we expected.

7.3 Zhongyu Wang

Most of time of the second half of this semester are devoted to the project. Painful thought, it really benefit me a lot.

Because we are designing a quite different scoping rule, it is quite hard to translate the code into C++. I got stuck several times, spend hours thinking it over and over, with no codes written. But finally when I make it, the strong sense of achievement swiped out all the unhappiness during the day and nights fighting with all the difficulties.

The project benefits me greatly by helping me thinking more about the principles in programming language. Why sub-function-declaration are not allowed by most of the languages? The answer might be a difficult realization of function call especially when functions can be conveyed from functions to functions. On the

other hand, OOP might make it not quite useful because it gives a convenient scoping rule.

Designing such a project is never as easy as I thought before. At last it is still not a perfect one. Optional parameters and function overload seems quite useful in our language. During the process I kept finding things needed to add and to modify. As a result, the whole structure of our program are not so clear and simple as it is first designed.

Finally, it is a project with several quite independent parts. It is important to design how to work on them simultaneously by different people, how to minimize changes in other part when something needs to be changed in one part.

The idea is originated from a famous software: geometer's sketchpad. All our design are visualized in the software. When I was preparing for the math competition in my high school it helped me a lot. The project reminds me of the sweet times solving annoying hard questions and discussing them with my friends.

7.4 Advice for future group

Every previous team would suggest starting early and so do we. But you don't need to rush to the codes, please consult your TA once submitted the Proposal and LRM for detailed feedback, because good design will save you whole bunch of time.

Do invite all the team member to the meeting or you will spend extra time explaining everything to the person who missed the last meeting. It hurts the efficiency.

In the earlier stage tasks should be divided to perform a parallel approach to the destination, however once most of the modules are finished, we suggest to do cross testing, so that everyone can have a comprehensive understanding of the whole project rather than limit his/her knowledge to the divided part. Also this method is good for finding bugs ignored by the original author.

As tech guys, you need to make full use of as much tools as possible. Tools like SVN, Google Docs, discuss groups, etc. will save you a lot of time and enable all group members to fully collaborate with each other. Though we do not keep a complete project log, but it's wise of you to do that. It's a good habit to keep everything neat and organized.

7.5 Special Thanks

Thanks Stephen for guiding us in fighting with the Dragon of compiler! Thanks our TA Mengqi Zhang to give us feedbacks and instructions in our project! Thanks authors of MicroC, Curve for providing us with an outline of how the project would be like!

8. Appendix

```
**************************************************************
scanner.mll Originally authored by Zhongyu Wang
**************************************************************

 { open Parser }


 rule token = parse
  [' ' '\t' '\r' '\n'] { token lexbuf }
 (*punctuations*)
 | '+' { PLUS } (*operators*)
 | '-' { MINUS }
 | '*' { TIMES }
 | '/' { DIVIDE }
 | '%' { MOD }
 | ">=" { LE }
 | "<=" { SE }
 | "==" { EQ }
 | "!=" {NEQ}
 | '>' { LARGER }
 | '<' { SMALLER }
 | '(' { LPAREN }
 | ')' { RPAREN }
 | '[' { LBRACK }
 | ']' { RBRACK }
```

```
| ';' {SEMICOLON}
| ',' {COMMA}
| '#' { COMMENT }
| '.' { DOLLAR}
| '=' { ASSIGN }
| "&&" { AND }
| "||" { OR }
| '~' { NOT }
| "void" {VOID}
| "integer" { INTEGER } (*types*)
| "string" { STRING }
| "float" { FLOAT }
| "bool" { BOOL }
| "Line" { LINE }
| "Circle" { CIRCLE }
| "Point" { POINT }
| "if" { IF }(*controlling sequence*)
| "else" { ELSE }
| "then" { THEN }
| "elif" { ELIF }
| "end" { END }
| "for" { FOR }
| "while" { WHILE }
| "break" { BREAK }
```

```
| "continue" { CONTINUE }
| "return" { RETURN }
| "pass" {PASS}
| "library" {LIB}
| "do" {DO}
| "true" { BOL(true) }
| "false"  { BOL(false) }
| ['0'-'9']+ as lit { INT(int_of_string lit) }
| ['0'-'9']*'.'['0'-'9']+ as lit { FLO(float_of_string lit) }
| '"'[^'"']*'"' as lit { STR(lit) }
| ['A'-'Z' 'a'-'z']+['A'-'Z' 'a'-'z' '0'-'9']* as lit { ID(lit) }
| eof { EOF }
| _  {raise (Failure("illegal character"))}
```

****

AST.mll Originally authored by Zhongyu Wang

****

```
type bioperator =
Add|Sub|Mul|Div|Mod|Le|Se|Eq|Neq|Larger|Smaller|And|Or(*binary
operators*)
type unioperator = Uminus|Not (*uniary operators*)
type vari_type=Int|Str|Flo|Bol|Lin|Cir|Poi|Void (*variable types*)

type expr = (*expressions*)
 Binop of expr * bioperator * expr (*a + 2*)
| Integer of int (*42*)
| Float of float(*3.2*)
| Bool of bool (*True*)
```

```
| String of string(*'abcdefg'*)
| Uniop of unioperator*expr(*-2 or ~True*)
| Assign of string*expr (*a=2*)
| Id of string (*identifiers*)
| Call of  string * expr list(*function call*)
| Property of string*string(*object property*)
| Property_Call of string*string*expr list
| Noexpr
| Library_Call of string*string*expr list

type var_req={(*where a variable is required, but not declared,like in formal list
return value of a function and *)
 v_type:vari_type;
v_name:string;
}

type var_decl={(*variable declaration*)
v_type:vari_type;
v_name: string;
init_value: expr;
 }

type stmt = (* Statements *)
Expr of expr
| Return of expr
| If of expr * stmt list * stmt list* stmt list
| For of expr * expr * expr * stmt list
| While of expr * stmt list
| Var_dec of var_decl
| Func_dec of func_decl
| Break
| Continue
| Pass
| Elif of expr * stmt list
and func_decl = {(*function declaration*)
```

```
return_type : var_req;
fname : string;
formals : var_req list;
body : stmt list; }
```

****

parser.mly Originally authored by Zhongyu Wang

****

```
%{
open Ast
(*f t returns the default initial value of type t*)
let f t =match t with
   Int -> Integer(0)
  |Str -> String("")
  |Flo -> Float(0.)
  |Bol -> Bool(false)
  | _ -> Integer(0)
%}
/*tokens got from scanner*/
%token LPAREN RPAREN LBRACK RBRACK LBRACE RBRACE
%token PLUS MINUS TIMES DIVIDE MOD ASSIGN
%token EQ LE SE NEQ LARGER SMALLER
%token AND OR NOT
%token SEMICOLON COMMA COMMENT DOLLAR
%token INTEGER STRING FLOAT BOOL LINE CIRCLE POINT VOID
%token RETURN IF ELSE ELIF FOR WHILE END BREAK CONTINUE DO THEN
%token PASS
%token LIB
%token <int> INT
%token <float> FLO
%token <string> STR
%token <string> ID
%token <bool> BOL
```

```
%token EOF


%right ASSIGN
%left AND OR
%left NOT
%left EQ NEQ
%left LE SE LARGER SMALLER
%left PLUS MINUS
%left UMINUS
%left TIMES DIVIDE MOD

%start program
%type < Ast.program> program

%%

/* Our program allows variable and function declaration any time*/
program:
  o_stmt {[$1]}
 | program o_stmt  { $2::$1 }

/*variable declaration*/
var_type:
   INTEGER {Int}
 | STRING  {Str}
 | FLOAT   {Flo}
 | BOOL    {Bol}
 | LINE    {Lin}
 | CIRCLE  {Cir}
 | POINT   {Poi}
 | VOID {Void}
v_decl:/*3 type of declarations*/
   var_type ID SEMICOLON { Var_dec({v_type=$1;v_name=$2;init_value=f
$1;}) }/*int a;*/
```

```
  | var_type ID ASSIGN expr SEMICOLON
{ Var_dec({v_type=$1;v_name=$2;init_value=$4;}) }/*int a=0*/
/*function declaration*/
f_decl:
   var_type ID LPAREN formals_opt RPAREN o_stmt_list END
{ Func_dec({ return_type={v_type=$1;v_name="";}; fname = $2; formals = $4;
body = List.rev $6 }) }
/*formals*/
formals_opt:
  /* nothing */ { [] }
  | formal_list { List.rev $1 }
formal_list:
    formal               { [$1] }
  | formal_list COMMA formal { $3 :: $1 }

formal:
   var_type ID { {v_type=$1;v_name=$2}}
/*expressions*/
expr:
  expr PLUS    expr { Binop($1, Add, $3) }
| expr MINUS   expr { Binop($1, Sub, $3) }
| expr TIMES   expr { Binop($1, Mul, $3) }
| expr DIVIDE  expr { Binop($1, Div, $3) }
| expr MOD     expr { Binop($1, Mod, $3) }
| expr EQ      expr { Binop($1, Eq,  $3) }
| expr NEQ     expr { Binop($1, Neq, $3) }
| expr LE      expr { Binop($1, Le,  $3) }
| expr SE      expr { Binop($1, Se,  $3) }
| expr LARGER  expr { Binop($1, Larger, $3) }
| expr SMALLER expr { Binop($1, Smaller, $3) }
| expr AND     expr { Binop($1, And, $3) }
| expr OR      expr { Binop($1, Or, $3) }
| ID ASSIGN expr {Assign($1,$3)}
| ID LPAREN actuals_opt RPAREN { Call($1, $3) }
| ID DOLLAR ID { Property($1,$3)}
```

```
| ID DOLLAR ID LPAREN actuals_opt RPAREN{Property_Call($1,$3,$5)}
| LIB DOLLAR ID LPAREN actuals_opt RPAREN{Library_Call("library",$3,$5)}
| LPAREN expr RPAREN { $2 }
| NOT expr {Uniop(Not,$2)}
| MINUS expr %prec UMINUS{Uniop(Uminus,$2)}
| INT        { Integer($1) }
| STR        { String($1)  }
| FLO        { Float($1)   }
| BOL        { Bool($1)    }
| ID         { Id($1)      }

expr_opt:
   /* nothing */ { Noexpr }
  | expr        { $1 }

/*actuals in function_all*/
actuals_opt:
/* nothing */ { [] }
| actuals_list { List.rev $1 }

actuals_list:
expr { [$1] }
| actuals_list COMMA expr { $3 :: $1 }

/* statements inside loops, ifs*/
stmt:
  expr SEMICOLON { Expr($1) }
| RETURN expr_opt SEMICOLON { Return($2) }
| BREAK SEMICOLON {Break}
| CONTINUE SEMICOLON {Continue}
| IF expr THEN stmt_list END {If($2,$4,[Pass],[Pass])}
| IF expr THEN stmt_list estmt_list END {If($2,$4,$5,[Pass])}
| IF expr THEN stmt_list estmt_list ELSE stmt_list END {If($2,$4,$5,$7)}
| IF expr THEN stmt_list ELSE stmt_list END {If($2,$4,[Pass],$6)}
```

```
| FOR expr_opt SEMICOLON expr_opt SEMICOLON expr_opt DO stmt_list END
{For($2,$4,$6,$8)}
| WHILE expr DO stmt_list  END {While($2,$4)}
| PASS SEMICOLON{Pass}
| v_decl{$1}
stmt_list:
   stmt_list stmt { $2 :: $1 }
  | stmt {[$1]}



 /*o_stmt allows variable and function declaration*/
o_stmt:
   stmt {$1}
  | f_decl {$1}


o_stmt_list:
   o_stmt_list o_stmt { $2 :: $1 }
  | o_stmt{[$1]}


/* else-if statement*/
estmt:
   ELIF expr THEN stmt_list { Elif($2,$4)}

estmt_list:
       estmt {[$1]}
  | estmt_list estmt {$2::$1}
```

****

semantic.ml Originally authored by Yichen Liu

****

```
open Ast


type env = {
        mutable functions : func_decl list;
        variables : (vari_type * string) list;
        }



(*this is a function used in other functions*)
(*It is used to test whether a function's name is equal to a string 'name'*)

let func_equal_name name = function
        | func -> func.fname = name


(*This function is to check whether a function's name has been defined
more than once*)

let fun_exist func env =
        let name = func.fname in
          try
                        let _ = List.find (func_equal_name name) env.functions
in
                                let e = "Function whose name is "^ name ^" has
been defined more than once" in
                                        raise (Failure e)
                with Not_found -> false

(*This function is to check whether a function's name exist in the env*)
let func_name_exist name env = List.exists (func_equal_name name)
env.functions
```

```
(*This function will directly give you the function object if you give its
name*)
let return_func_given_name name env =
        try
                let result = List.find (func_equal_name name) env.functions
in
                        result
 with Not_found -> raise(Failure("Function "^ name ^ " has not been
declared!"))

(*check whether a 'fpname' is in the formal parameter list of a function*)
let exist_formal_para func fpname =
        let check func fpname = List.exists (function (a,b) -> b = fpname)
func.formals in
         check func fpname

(*check whether a 'vname' is declared in the local variable list of a
function*)

let exist_local_variable func vname =
   let check func vname = List.exists (function (a,b) -> b= vname) func.locals
in
      check func vname


(*check whether a 'vname' is declared in the global variable list*)

let exist_global_variable env vname =
        let check env vname = List.exists (function (a,b) -> b= vname)
env.variables in
         check env vname

(*get the type of a 'vname' in the global variable list*)

let get_global_variable_type env vname =
```

```
        try
                let fpara = List.find(function (_,b) -> b = vname) env.variables
in
                  let(c,_) = fpara in
                          c
        with Not_found ->raise(Failure("Cannot find a variable called "^
vname ^ " in global variable list!"))

(*The following function will get you the type of a parameter in a function if
you give parameter's name*)

let get_para_type func fpname =
        try
                  let para  =List.find (function (a,b) -> b = fpname) func.formals
in (*check whether fname is in  func's formal_list*)
                          let (para_type,_) = para in
                                  para_type
        with Not_found ->raise(Failure("In the function" ^ func.fname ^"
does not have the parameter "^ fpname))




(*A function to check whether a function has a parameter called fpname*)
let check_exist_para_in_fun func fpname = List.exists (function (_,b)->b =
fpname) func.formals


(*A function to check whether a function has a local parameter called
vname*)
let check_exist_var_in_fun func vname = List.exists (function (_,b) -> b =
vname) func.locals
```

```
(*The function to get the type of a variable "name" to see whether it
appears in parameter or local variable list or global variable list*)

let get_type env func name =
        if check_exist_var_in_fun func name
             then get_var_type func name
        else
                  if check_exist_para_in_fun func name
                               then get_para_type func name
                          else
                                  if exist_global_variable env name
                                          then get_global_variable_type env
name
                                          else
                                   raise(Failure("Variable" ^ name^" is used
but not declared in function "^ func.fname))


(*The function will check whether a variable "name" exists in a function's
para list or local list*)
let exist_id name func env= (check_exist_para_in_fun func name) or
(check_exist_var_in_fun func name) or (exist_global_variable env name)


(*The function will check whether a function name "fun_name" has a
corresponding function in the environment*)

let find_func func_name env =
        try
                let _ = List.find (func_equal_name func_name) env.functions
in
                   true
        with Not_found -> raise(Failure("Function "^ func_name ^" is not
found in the function defination list"))
```

```
(*This function will check whether a (var_type*string) has a para_name
that appears more than once in a function's parameter list*)
let count_fpara func = function (_,b)
  -> let f count (_,c) =
                  if c=b then count+1
                        else count
                        in
                          let count = List.fold_left f 0 func.formals in
                            if count > 1
                                    then raise(Failure("Duplicate parameter in
function " ^ func.fname))
                                    else
                                            count


(*This function will automatically check whether there is parameter
duplication in function defination*)

let check_fpara_duplicate func =
        List.map (count_fpara func) func.formals

(*This function will check whether a (var_type*string) has a var_name that
appears more than once in a function's local variable list*)
let count_var func = function (_,b)
  -> let f count (_,c) =
                  if c=b then count+1
                        else count
                        in
                          let count = List.fold_left f 0 func.locals in
                            if count > 1
                                    then raise(Failure("Duplicate parameter "^
b ^ " in function " ^ func.fname))
                                    else
                                            count
```

```
(*This function will automatically check whether a function has local
variable duplication*)

let check_var_duplicate func =
        List.map (count_var func) func.locals




(*This function will get you the return type of a function if you give me the
fname*)
let get_func_return_type func_name env =
        try
                let func = List.find (func_equal_name func_name)
env.functions in
                    func.ftype
        with Not_found -> raise(Failure("Function "^ func_name ^" is not
found in the function defination list"))



(*This function will return the type of an expression*)
(*env: the environment including the global variable list and function list*)
(*expr: the expression we are exploring*)
(*func:the function we are in now*)
let rec get_expr_type expr func env =
        match expr with
        | STR(s) ->Str
        | ID(s) -> get_type env func s (*we need to modify it later*)
        | INT(s) -> Int
        | FLO(s) ->Flo
```

```
      | Not(expr1) -> let type1 = get_expr_type expr1 func env in if type1 =
Bol then Bol else raise(Failure("The type of expression in Not operator
should be boolean!"))
        | Binop(expr1,op,expr2) -> let temp1 = get_expr_type expr1 func env
and temp2 = get_expr_type expr2 func env
            in
                          begin
                                match temp1,op,temp2 with
                        | Bol, Or, Bol-> Bol (*Or*)
                        | Bol, And, Bol -> Bol (*And*)
                                | Bol, Eq, Bol -> Bol(*Equal*)
                                | Int, Eq, Int -> Bol
                                | Int, Eq, Flo -> Bol
                                | Flo, Eq, Int -> Bol
                                | Flo, Eq, Flo -> Bol
                                | Lin, Eq, Lin -> Bol
                                | Poi, Eq, Poi -> Bol
                                | Bol, Neq, Bol->Bol (*Neq*)
                                | Int, Neq, Int -> Bol
                                | Int, Neq, Flo -> Bol
                                | Flo, Neq, Int -> Bol
                                | Flo, Neq, Flo -> Bol
                                | Cir, Neq,    Cir -> Bol
                                | Lin, Neq, Lin -> Bol
                                | Poi, Neq, Poi -> Bol
                                | Str, Neq, Str-> Bol
                                | Int, Se, Int -> Bol   (*Se*)
                                | Int, Se, Flo -> Bol
                                | Flo, Se, Int -> Bol
                                | Flo, Se, Flo -> Bol
                               | Int, Le, Int -> Bol   (*Le*)
                                | Int, Le, Flo -> Bol
                                | Flo, Le, Int -> Bol
                                | Flo, Le, Flo -> Bol
                                | Int, Smaller, Int -> Bol   (*Smaller*)
```

```
                                    | Int, Smaller, Flo -> Bol
                                    | Flo, Smaller, Int -> Bol
                                    | Flo, Smaller, Flo -> Bol
                                    | Int, Larger, Int -> Bol   (*Larger*)
                                    | Int, Larger, Flo -> Bol
                                    | Flo, Larger, Int -> Bol
                                    | Flo, Larger, Flo -> Bol
                                    | Int, Add, Int -> Bol   (*Add*)
                                    | Int, Add, Flo -> Bol
                                    | Flo, Add, Int -> Bol
                                    | Flo, Add, Flo -> Bol
                                    | Int, Sub, Int -> Bol   (*Sub*)
                                    | Int, Sub, Flo -> Bol
                                    | Flo, Sub, Int -> Bol
                                    | Flo, Sub, Flo -> Bol
                                    | Int, Mul, Int -> Bol   (*Mul*)
                                    | Int, Mul, Flo -> Bol
                                    | Flo, Mul, Int -> Bol
                                    | Flo, Mul, Flo -> Bol
                                    | Int, Div, Int -> Bol   (*Div*)
                                    | Int, Div, Flo -> Bol
                                    | Flo, Div, Int -> Bol
                                    | Flo, Div, Flo -> Bol
                                | Int, Mod, Int -> Bol   (*Mod*)
                                    | _,_,_ ->raise(Failure("Illegal type used in a
binop expression"))
                                    end(*end of binop consideration*)
                                |Assign(id,expr) -> get_expr_type expr func env
                                |Call(fname,expr) -> get_func_return_type fname
env(*We want to check the return type of the function fname*)
                                |_ ->raise(Failure("An unexpected error occurred!"))


(*The following function check whether a statements list end with a return
statement*)
```

```
let has_return_stmt stmt_list =
        if List.length stmt_list = 0
         then false
        else match (List.hd (List.rev stmt_list)) with
        | Return(_) -> true
        | _->false

(*Check is a statement list, whether one if statements has a return value in
it*)
let if_has_return stmt_list =
        let if_stmt_list = List.filter (function If(_,_,_)->true |_ ->false)
stmt_list in
          let check_result = List.map (
                    function
                            If(_,s1,s2) ->
                                    begin
                                                match s1,s2 with
                                                | Block(st1), Block(st2) -
>(has_return_stmt st1) && (has_return_stmt st2)
                                                | _ -> raise(Failure("The
statements after if or else should be included in {}!"))
                                                end
                     |_ -> false
                    ) if_stmt_list in(*get a list "check_result" of boolean to
tell whether each if statement has a return *)
                    List.fold_left (fun a b -> b||a) false check_result


(*The following function will judge whether an expression is assign or call*)

let is_assign_call func = function
        | Assign(_,_) ->true
        | Call(_,_) ->true
        | _ ->false
```

```
(*The following function will judge whether an expression is valid in a fund
and expr*)

let rec expr_valid func expr env =
        match expr with
        |Assign(id, e1) -> if exist_id id func env
                        then let type1 = get_type env func id and _ = expr_valid
func e1 env and type2 = get_expr_type e1 func env in

        if type1 = type2 then true

        else

        begin

        match type1, type2 with

        | Int,Flo -> true

        | Flo,Int ->true

        | _ ->raise(Failure"Unmathed type in Assign operation!")

        end
                                                                        else
raise(Failure("Undeclared identifier " ^ id ^ " is used!"))
        |Call(fname,exprlist) ->
                    if func_name_exist fname env
                                        then let _fulfill_valid_exprs = List.map
(fun e -> expr_valid func e env) exprlist in
                                                let _check_type =
check_func_paralist_type fname exprlist func env in
                                                        true
```

```
                                    else raise(Failure("Undefined function: "^
fname ^ "is used!"))
        |_ ->
                try
                        let _ = get_expr_type expr func env in true
                with Not_found -> raise(Failure("Invalid expression!"))


(*The following function will tell you whether a function's body is valid*)

let check_valid_body func env =
  let rec check_stmt =
                function
                        |Break->true
                        |Continue->true
                      |Return(expr) -> let ret  = get_expr_type expr func env in
                                        let real_type  = func.ftype in

                if ret = real_type then true else raise(Failure("Unmatched
return type with function's definition!"))
                        |If(expr,stmt1,stmt2) ->
                                        let expr_type = get_expr_type expr func
env in
                                          let _check_expr_type =
                                                  begin
                                                    match expr_type with
                                                        | Bol -> true
                                                        | _ ->
raise(Failure("expression in If(..) should be type BOOLEAN!"))
                                                  end
                                                        in
                                                        if (check_stmt stmt1) &&
(check_stmt stmt2)
                                                        then true
```

```
                                                    else raise(Failure("Invalid
statement in the if statement in function: "^ func.fname))
                                |For(a,b,c,stmt1) -> if expr_valid func a env &&
expr_valid func b env && expr_valid func c env && check_stmt stmt1 then
true else raise(Failure("Invalid statement or expressions in For statement in
function:"^ func.fname))
                                |While(a,stmt1)-> if expr_valid func a env &&
check_stmt stmt1 then true else raise(Failure("Invalid statement in While
statement in function:"^ func.fname))
                                in (*end of check_stmt*)
    let _ = List.map (check_stmt) func.body in
                        true


(*check whether a function's body has a return statement*)
let func_body_has_return func =
        let stmt_list = func.body in
          let result = List.exists (function stmt -> match stmt with |Return(_) -
>true |_ ->false) stmt_list          in
                  result
(*get all the return statements' corresponding expression type*)
let fun_get_all_return_type func env=
        let stmt_list = func.body in
         let f result_list stmt =
                match stmt with
                |Return(expr) -> (get_expr_type expr func env)::result_list
                |_ -> result_list in
          let result = List.fold_left f [] stmt_list in
                  result

(*check whether a function's return statement's type all fulfills the real
return type of this function*)
let check_return func env =
        match func.ftype with
```

```
        |  Void -> if func_body_has_return func then raise(Failure("There
should not be return statement in a void function: "^ func.fname)) else true
        |  f_type ->
                let return_type_list  =  fun_get_all_return_type func env in
            let f a  =  a= f_type in
                                if List.for_all  f return_type_list  then true else
raise(Failure("At least one return type does not match the function's
definition requirement!"))

(*THis will check each function's validity*)
let check_func f env =
                let _dup_name = fun_exist f env in
              let _ = env.functions <- (f) ::env.functions in
                  let _dup_formals = check_fpara_duplicate f in
                        let _dup_vlocals = check_var_duplicate f in
                              let _vbody = check_valid_body f env in
                                    let _check_return_result = check_return f
env in

            true



(*the following three functions will judge whether there is global variables
redefiend!*)
let equal_variable_name (a,b) (c,d) =
        b=d

let exist_v_name vlist vdecl =
        let new_fun count x =
                if(equal_variable_name vdecl x) then count+1 else count in
                 let result = List.fold_left new_fun 0 vlist in
                   if result <=1 then true else raise(Failure("Global Variable has
been redefined!"))
```

```
  let dup_in_global env =
          List.for_all (exist_v_name env.variables) env.variables



  let check_program (var_list,fun_list) =
          let env = {functions = built_in;variables = var_list} in
        let _global_check = dup_in_global env in
          let _dovalidation = List.map (fun f -> check_func f env) fun_list in
                  let _ = print_endline "\nThe semantic check has been
  finished!\n" in
                          true
```

****

codegen.ml Originally authored by Zhongyu Wang

****

```
open Ast

let block_start="Symbol_Table_Node*
parent=_CurSymbolTable;\nSymbol_Table_Node* _CurSymbolTable=new
Symbol_Table_Node(parent);\n"
let block_end="delete _CurSymbolTable;\n"
(*types used to construct a environment*)
type record={name:string;start_str:string;end_str:string;var_type:vari_type;}
type env={global:record list;local:record
list;func_count:int;func_code:string;is_global:bool;}

(*below functions are used for searching through environments*)
let rec search_env_list var env_list=match env_list with
  h::t->if h.name=var then h.name,h.start_str,h.end_str,h.var_type,true else
search_env_list var t
  |[]->"@","@","@",Void,false

let search_var var env=(*search variables in environment*)
```

```
  let a,b,c,d,e=search_env_list var env.local in
          if e=false then (search_env_list var env.global) else a,b,c,d,e
and add_var var env=(*add variables to environment*)
  if env.is_global=true then
{global=var::env.global;local=env.local;func_count=env.func_count;func_cod
e=env.func_code;is_global=env.is_global}
  else
{global=env.global;local=var::env.local;func_count=env.func_count;func_cod
e=env.func_code;is_global=env.is_global}

let concat a b= a^b
let str_of_op= function(*convert operators to string*)
  Add->"+"
  |Sub->"-"
  |Mul->"*"
  |Div->"/"
  |Mod->"%"
  |Eq->"=="
  |Neq->"!="
  |Le->">="
  |Se->"<="
  |Larger->">"
  |Smaller->"<"
  |And->"&&"
  |Or->"||"
let str_of_vartype=function(*convert types to string*)
  Int->"int"
  |Str->"string"
  |Flo->"double"
  |Bol->"bool"
  |Lin->"Line*"
  |Cir->"Circle*"
  |Poi->"Point*"
  |Void->"void"
let str_of_vartype_2=function(*convert types to string*)
```

```
  Int->"int"
  |Str->"string"
  |Flo->"double"
  |Bol->"bool"
  |Lin->"Line*"
  |Cir->"Circle*"
  |Poi->"Point*"
  |Void->"void"
let str_of_vartype_call=function(*convert types to string*)
  Int->"*((int*"
  |Str->"*((string*"
  |Flo->"*((double*"
  |Bol->"*((bool*"
  |Lin->"(*(Line**"
  |Cir->"(*(Circle**"
  |Poi->"(*(Point**"
  |Void->"((void"
let rec eval_stmt stmt env=match stmt with
  Expr(e)->(eval e env)^";\n",env
  |Return(e)-> "_return_value="^(eval e env)^";\n goto endline;\n",env
  |If(e,s1,s2,s3)->"if("^(eval e env)^"){\n"^(let x,y =eval_stmt_list (List.rev s1)
env in block_start^x^block_end)^"}\n"^(let x,y =eval_stmt_list (List.rev s2)
env in x)^"else{\n"^(let x,y =eval_stmt_list (List.rev s3) env in
block_start^x^block_end)^"}\n",env
  |For(e1,e2,e3,s1)->"for("^(eval e1 env)^";"^(eval e2 env)^";"^(eval e3
env)^"){\n"^(let x,y =eval_stmt_list (List.rev s1) env in
block_start^x^block_end)^"}\n",env
 |While(e,s)->"While("^(eval e env)^"){\n"^(let x,y =eval_stmt_list (List.rev s)
env in block_start^x^block_end)^"}\n",env
  |Var_dec(v)->let x,y=(eval_var_dec v env) in x^";\n",y
  |Func_dec(f)->let x,y=(eval_fun_dec f env)in x^"\n",y
  |Break-> "break;\n",env
  |Continue->"continue;\n",env
  |Pass->"\n",env
```

```
  |Elif(e,s)->"else if("^(eval e env)^"){\n"^block_start^(let x,y =eval_stmt_list
(List.rev s) env in x)^block_end^"}\n",env
and eval_stmt_list stmt_list env=match stmt_list with
  h::t->let code1,env1 =eval_stmt h env in let code2,env2 = eval_stmt_list t
env1 in code1^code2, env2
  (*|[_]->eval_stmt (List.hd stmt_list) env*)
  |_->"",env
and eval_fun_dec func env=(**)
  let
fun_env={global=env.global;local=[];func_count=env.func_count;func_code=
env.func_code;is_global=false} in(*create a new environment for function*)
  let var_req_str=eval_var_req func.return_type
  and f_name=(ignore (print_endline func.fname));func.fname(*function
name*)
  and str_dec,str_poi,str_head,new_env=eval_actual func.formals fun_env
in(*str_dec:(int a, int b) str_poi:(int,int) str_head: symbol_table->add_var()*)
  let
last_env={global=new_env.global;local={name=f_name;start_str="("^var_req
_str^"(*";end_str=")("^str_poi^"))";var_type=Int;}::new_env.local;func_coun
t=new_env.func_count+1;func_code=new_env.func_code;is_global=new_en
v.is_global}
  in let stmts,env2=eval_stmt_list  func.body last_env
  in "_CurSymbolTable->add_function((func_pointer)
&function"^(string_of_int (new_env.func_count+1))^",\""^f_name^"\");\n",
  {global=if env.is_global then
{name=f_name;start_str="("^var_req_str^"(*";end_str=")("^str_poi^"))";var_
type=Int;}::env.global else env.global;
    local=if env.is_global then env.local else
{name=f_name;start_str="("^var_req_str^"(*";end_str=")("^str_poi^"))";var_
type=Int;}::env.local;
  func_count=env2.func_count;
  func_code=env2.func_code^var_req_str^" function"^(string_of_int
(new_env.func_count+1))^"("^str_dec^"){\n"^(if
func.return_type.v_type!=Void then(str_of_vartype
func.return_type.v_type)^" _return_value;" else
```

```
"")^block_start^str_head^"_CurSymbolTable->add_function((func_pointer)
&function"^(string_of_int
(new_env.func_count+1))^",\""^f_name^"\");\n"^stmts^"endline:\n"^block_
end^(if func.return_type.v_type!=Void then "return _return_value;" else "")
^"}\n";
  is_global=env.is_global}
  (*(eval_var_req func.return_type)^" "^func.fname^"("^(eval_actual
func.formals)^"){\n"^(List.fold_left concat "" (List.map eval_stmt
func.body))^"}"*)
and eval_var_req vr=
  (str_of_vartype vr.v_type)^" "^vr.v_name
and eval_var_dec vd env=(**)
  "_CurSymbolTable->add_var("^(eval_vardec_init vd
env)^",\""^vd.v_name^"\")",{global=if env.is_global then
{name=vd.v_name;start_str="";end_str="";var_type=vd.v_type}::env.global
else env.global;local=if env.is_global then env.local else
{name=vd.v_name;start_str="";end_str="";var_type=vd.v_type}::env.local;fu
nc_count=env.func_count;func_code=env.func_code;is_global=env.is_global}

and eval expr env = match expr with
  Binop(e1,op,e2) -> "("^(eval e1 env)^(str_of_op op)^(eval e2 env)^")"
  | Call(id,formal)-> let name,str_start,str_end,v_type,is_found =search_var
id env in "("^str_start^str_end^"_CurSymbolTable-
>get_function(\""^id^"\"))"^"("^(eval_formal formal env)^")"
  (*id^"("^(eval_formal formal env)^")"*)(**)
  | Property(id,prop)->let name,str_start,str_end,v_type,is_found
=search_var id env in "(("^(str_of_vartype v_type)^")_CurSymbolTable-
>get_var(\""^id^"\"))"^"->"^prop(**)
  |Property_Call(id,method_name,paras)->let
name,str_start,str_end,v_type,is_found =search_var id env in let parameters
= eval_formal paras env in ""^(str_of_vartype_call
v_type)^")_CurSymbolTable->get_var(\""^id^"\"))"^"-
>"^method_name^"("^parameters^")"(**)
  | Integer(value)-> string_of_int value
  | String(value)->value
```

```
  | Float(value)->string_of_float value
  | Bool(value)->string_of_bool value
  | Id(id)->let name,str_start,str_end,v_type,is_found =search_var id env in
(*"*(("^*)(str_of_vartype_call v_type)^")_CurSymbolTable-
>get_var(\""^id^"\"))"(*id*)(*need to get type!*)
  | Assign(id, e)->let name,str_start,str_end,v_type,is_found =search_var id
env in "*(("^(str_of_vartype_2 v_type)^"*)_CurSymbolTable-
>get_var(\""^id^"\"))="^(eval e env)
  | Noexpr->""
  | Uniop(op,e)->(if op==Uminus then "-" else "!")^(eval e env)
  | Library_Call(lib,method_name,paras)-
>"library."^method_name^"("^(eval_formal paras env)^")"
and eval_program pr env=
  eval_stmt_list (List.rev pr) env
and eval_actual actual_list env=match actual_list with(*evaluate actuals and
give its form*)
  []->"","","",env(*(int a,int b),(int,int),(_CurSymbolTable->add_var))*)
  |[_]->(eval_var_req (List.hd actual_list)),(str_of_vartype (List.hd
actual_list).v_type),"_CurSymbolTable->add_var("^(List.hd
actual_list).v_name^",\""^(List.hd
actual_list).v_name^"\");\n",{global=env.global; local={name=(List.hd
actual_list).v_name;start_str="";end_str="";var_type=(List.hd
actual_list).v_type}::env.local;func_count=env.func_count;func_code=env.fu
nc_code;is_global=false}
  |h::t -> let x1,y1,z1,w1 =(eval_actual [h] env) in let x,y,z,w =(eval_actual t
w1) in  x1^","^x,y1^","^y,z1^z,w
and eval_formal formal_list env=match formal_list with(**)
  []->""
  |[_]->(eval (List.hd formal_list) env)
  |h::t -> (eval h env)^","^(eval_formal t env)
and eval_vardec_init vd env= match vd with(**)
  {v_type=Cir;init_value=Integer(0)}->"NULL"(*"new Circle"*)
  |{v_type=Lin;init_value=Integer(0)}->"NULL"(*"new Line"*)
  |{v_type=Poi;init_value=Integer(0)}->"NULL"(*"new Point"*)
  |_->(eval vd.init_value env)
```

```
and eval_count v = match v with
  |_->v

(*main program*)

let _ =
 let head="int main(){\n_sketch=new Sketch;_CurSymbolTable=new
Symbol_Table_Node(NULL);\nfopen_s(&_resultfile,\"result.txt\",\"w\");"
 and head1="#include \"stdafx.h\"\n#include <string>\n#include
<iostream>\n#include <list>\nusing namespace std;\nlib library;\nSketch*
_sketch;\nSymbol_Table_Node* _CurSymbolTable;\nFILE* _resultfile;\n"
 and tail=block_end^"}"
 and env={global=[];local=[];func_count=0;func_code="";is_global=true}
 and file_out = open_out "result.cpp"
 and file = open_in "a.txt" in
 let lexbuf = Lexing.from_channel file in
 let program = Parser.program Scanner.token lexbuf in
 let result,newenv =  eval_program program env in
 ignore ""(*print_endline result*);ignore (output_string file_out
(head1^newenv.func_code^head^result^tail));ignore (close_out file_out);;
(*input_line stdin*)
```

****

stdafx.h Originally authored by Yan Peng

****

```
// stdafx.h : include file for standard system include files,
// or project specific include files that are used frequently, but
// are changed infrequently
//
#pragma once
```

```
#include "targetver.h"
#include <string>
#include <iostream>
#include <math.h>
#include <stdio.h>
using namespace std;
#include <stdio.h>
#include <tchar.h>
const double NONEXIST=-65536.;
class GeoObj;
class Line;
class Point;
class Circle;
class ChildList;
enum det_type{//determine type
	FREE,
	CIR_NEAR,
	CIR_FAR,
	LIN,
	START_END,
	CENTER_RADIUS,
	CENTER
};
enum lin_type{//line type
```

```
        SEGMENT=0,
        RAY=1,
        LINE=2
};
enum geo_type{
        GLINE,
        GCIRCLE,
        GPOINT
};
struct ChildNode{//used in ChildList
        struct ChildNode * pre;
        struct ChildNode * next;
        GeoObj* value;
};

// TODO: reference additional headers your program requires here
class ChildList{//ChildList is used to record all childrens of a geo-obj
public:
        ChildList();
        struct ChildNode* start;
        struct ChildNode* end;
        int num;
        void add(GeoObj* value);
        void del(GeoObj* node);
```

```
        ChildNode* find(GeoObj* node);
        ~ChildList();
};
struct Node{
        GeoObj* value;
        Node* next;
};
class List{
public:
        Node* start;
        Node* end;
        int num;
        List();
        void add(GeoObj* GeoObj);
        void del(GeoObj* Obj);
};
class Sketch{
public:
        double height;
        double width;
        List* eleList;
        Sketch();
        void paint();
};
```

```
extern Sketch* _sketch;
extern FILE* _resultfile;
class GeoObj{
public:
	det_type determine_type;//how the object is determined
	string name;//name shown on the graph
	int color;//color shown on the graph
	int parentnum;//parentnums,0 for none, 1 for only the first, 2 for only the second, 3 for both
	bool visible;//whether it is shown on the graph
	GeoObj* parent[2];//the parents of the object
	ChildList* children;//the children of the object
	Sketch* sketch;//the sketchpad it belongs to
	geo_type type;//the type of the object: line,circle or point
	bool visited;//used to update graph
	int getcolor();
	int setcolor(int color);
	bool isvisible();
	bool setvisible(bool visible);
	void setname(string name);
	string getname();
	virtual bool update(void)=0;
	virtual void paint(void)=0;
	virtual bool hasnon(void)=0;
	void unlock();
```

```
        GeoObj(string name="",bool isvisible=true);
        static double* intercept(GeoObj* obj1,GeoObj* obj2);
        ~GeoObj();
};
class Point:public GeoObj{
protected:
        double X;
        double Y;
public:
        friend class GeoObj;friend class Line;friend class Point;friend class Circle;
        Point();
        static double dist(Point* point1, Point* point2);
        static Point* intersect(GeoObj* GeoObj1,GeoObj* GeoObj2,bool isnear);
        bool hasnon(void);
        bool update(void);
        bool setXY(double X,double Y);
        double getX();
        double getY();
        void paint();
};
class Line:public GeoObj{
protected:
        double X1,Y1,X2,Y2;
public:
```

```
        friend class GeoObj;friend class Line;friend class Point;friend class Circle;
        lin_type line_type;
        Line();
        bool update(void);
        double length();
        static Line* LineXY(Point* P1,Point* P2,lin_type line_type=SEGMENT);
        double getX1();
        double getY1();
        double getX2();
        double getY2();
        Point* getStart();
        Point* getEnd();
        bool setP1(double X1,double Y1);
        bool setP2(double X2,double Y2);
        void paint();
        bool hasnon(void);
        int getLineType();
};
class Circle:public GeoObj{
protected:
        double X,Y;
        double radius;
public:
        friend class GeoObj;friend class Line;friend class Point;friend class Circle;
```

```
        Circle();
        static Circle* DrawCircle(Point* center, Line* radius);
        static Circle* DrawCircle(Point* center, double radius);
        bool update(void);
        double getX();
        double getY();
        double getRadius();
        bool setCenter(double X,double Y);
        bool setRadius(double r);
        Point* getCenter();
        void paint();
        bool hasnon(void);
};


typedef void(*func_pointer)();
enum type{
        var_num,
        var_float,
        var_string,
        var_bool,
        var_point,
        var_circle,
        var_line
```

```
};
class variable{
public:
	variable();
	type var_type;
	union {
		int* val_int;
		double* val_float;
		bool* val_bool;
		string* val_string;
		Point* val_point;
		Circle* val_circle;
		Line* val_line;
	}var_strorage;
	variable* next;
	string name;
};
class function{
public:
	func_pointer funcp;
	function* next;
	string name;
};
class Symbol_Table_Node{
```

```
public:
        Symbol_Table_Node* parent;
        variable* First;
        variable* Last;
        function* Firstf;
        function* Lastf;
        bool operator!=(const Symbol_Table_Node &orig)const;
        Symbol_Table_Node(Symbol_Table_Node* parent);
        ~Symbol_Table_Node();
        Symbol_Table_Node* add_node();
        variable* add_var(int value,string name);
        variable* add_var(string value,string name);
        variable* add_var(double value,string name);
        variable* add_var(bool value,string name);
        variable* add_var(Point* value,string name);
        variable* add_var(Circle* value,string name);
        variable* add_var(Line* value,string name);
        void add_function(func_pointer funcp,string name);
        void* get_var(string name,bool is_local=false);
        func_pointer get_function(string name,bool is_local=false);
};
//************************************
class lib{
public:
```

```
        static int print(string a);
        static double sin(double x);
        static Line* LineST(Point* P1,Point* P2,int line_type);
        static Circle* DrawCircle(Point* center, Line* radius);
        static Circle* DrawCircle(Point* center, double radius=NONEXIST);
        static Point* intersect(GeoObj* GeoObj1,GeoObj* GeoObj2,bool isnear);
        static void paint();
        Point* lib::PointXY(double X,double Y);
        Line* LineXY(double X1,double Y1,double X2,double Y2,int line_type);
        Circle* CircleXYr(double X,double Y,double r);
};
```

1. stdafx.cpp

```
// stdafx.cpp : source file that includes just the standard includes
// ConsoleApplication2.pch will be the pre-compiled header
// stdafx.obj will contain the pre-compiled type information
#include "stdafx.h"
// TODO: reference any additional headers you need in STDAFX.H
// and not in this file
        ChildList::ChildList(){
                start=NULL;
                end=NULL;
                num=0;
        }
```

```
void ChildList::add(GeoObj* value){
	struct ChildNode* newnode=new ChildNode;
	newnode->value=value;
	if (this->num==0){
		this->start=newnode;
		this->end=newnode;
	}
	else{
		this->end->next=newnode;
		newnode->next=NULL;
		newnode->pre=this->end;
		this->end=newnode;
	}
	this->num+=1;
}
void ChildList::del(GeoObj* node){
	ChildNode* PNodeToDel=find(node);
	if (PNodeToDel==NULL){
		return;
	}
	else{
		if (this->num==1){
			delete this->start;
			this->start=NULL;
```

```
                this->end=NULL;
            }
            else{
                this->end->pre->next=NULL;
                ChildNode* cur=this->end;
                this->end=this->end->pre;
                delete cur;
            }
            this->num-=1;
        }
}
ChildNode* ChildList::find(GeoObj* node){
        ChildNode* current;
        current=this->start;
        while (current!=NULL){
            if (current->value==node){
                return current;
            }
            current=current->next;
        }
        return NULL;
}
ChildList::~ChildList(){
        ChildNode* current=this->start;
```

```
        while (current!=NULL){
                ChildNode* temp=current->next;
                delete current;
                current=temp;
        }
}
List::List(){
        this->start=NULL;
        this->end=NULL;
        this->num=0;
}
void List::add(GeoObj* GeoObj){
        Node* N=new Node;
        N->value=GeoObj;
        N->next=NULL;
        if (num==0){
                this->num++;
                this->start=end=N;
        }
        else{
                this->num++;
                this->end->next=N;
                this->end=N;
        }
```

```
}
void List::del(GeoObj* Obj){
	Node* current=this->start,*pre=NULL;
	while (current!=NULL){
		if (current->value==Obj){
			if (pre==NULL){
				this->start=current->next;
				delete current;
			}
			else{
				pre->next=current->next;
				delete current;
			}
			this->num--;
			return;
		}
	}
}
Sketch::Sketch(){
	eleList=new List;
}
void GeoObj::unlock(){
	if (this->parentnum==0){
		return;
```

```
        }
        if (this->parentnum%2==1){
                this->parent[0]->children->del(this);
                this->parent[0]=NULL;
        }
        if (this->parentnum>=2){
                this->parent[1]->children->del(this);
                this->parent[1]=NULL;
        }
        this->determine_type=FREE;
}
GeoObj::GeoObj(string name,bool isvisible){
        this->name=name;
        this->visible=isvisible;
        this->color=0x000000;
        this->parentnum=0;
        this->parent[0]=NULL;
        this->parent[1]=NULL;
        this->visited=false;
        this->children=new ChildList;
        this->sketch=_sketch;
        this->determine_type=FREE;
        this->sketch->eleList->add(this);
}
```

```
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);
}
bool GeoObj::setvisible(bool visible){
	return this->visible=visible;
}
Point::Point():GeoObj(){
	this->X=NONEXIST;
	this->Y=NONEXIST;
	this->type=GPOINT;
}
double Point::dist(Point* point1, Point* point2){
	return sqrt((point1->X-point2->X)*(point1->X-point2->X)+(point1->Y-
point2->Y)*(point1->Y-point2->Y));
}
Point* Point::intersect(GeoObj* GeoObj1,GeoObj* GeoObj2,bool isnear){
```

```
        if (GeoObj1->type==GPOINT || GeoObj2->type==GPOINT||GeoObj1->sketch!=GeoObj2->sketch) return NULL;
        Sketch* sketch=GeoObj1->sketch;
        if (GeoObj1->type==GCIRCLE || GeoObj2->type==GCIRCLE){
            Point* interpoint=new Point;
            if (isnear){
                interpoint->parent[0]=GeoObj1;
                interpoint->parent[1]=GeoObj2;
                interpoint->determine_type=CIR_NEAR;
                GeoObj1->children->add(interpoint);
                GeoObj2->children->add(interpoint);
                interpoint->update();
            }
            else{
                interpoint->parent[0]=GeoObj1;
                interpoint->parent[1]=GeoObj2;
                interpoint->determine_type=CIR_FAR;
                GeoObj1->children->add(interpoint);
                GeoObj2->children->add(interpoint);
                interpoint->update();
            }
            return interpoint;
        }
        else{
            Point* interpoint=new Point;
```

```
                interpoint->parent[0]=GeoObj1;
                interpoint->parent[1]=GeoObj2;
                GeoObj1->children->add(interpoint);
                GeoObj2->children->add(interpoint);
                interpoint->determine_type=LIN;
                interpoint->update();
                return interpoint;
        }
    }
    bool Point::hasnon(void){
        if (this->X==NONEXIST||this->Y==NONEXIST){
            return true;
        }
        return false;
    }
    bool Point::update(void){
        if (this->determine_type==FREE){
            return false;
        }
        else{
            double* result=GeoObj::intercept(this->parent[0],this-
>parent[1]);
            if (this->determine_type==CIR_FAR){
                this->X=result[2];
                this->Y=result[3];
```

```
            }
            else if (this->determine_type==CIR_NEAR){
                this->X=result[0];
                this->Y=result[1];
            }
            else if (this->determine_type==LIN){
                this->X=result[0];
                this->Y=result[1];
            }
        }
        ChildNode* current=this->children->start;
        while (current!=NULL){
            current->value->update();
        }
        return true;
}
bool Point::setXY(double X,double Y){
        if(this->determine_type==FREE){
            this->X=X;
            this->Y=Y;
            this->update();
            return true;
        }
        else{
```

```
                return false;
        }
    }
    double Point::getX(){
        return this->X;
    }
    double Point::getY(){
        return this->Y;
    }

    void Point::paint(){
        if (this->visible && !this->hasnon()){
            fprintf(_resultfile,"Point %f %f %s\n",this->X,this->Y,this->name.c_str());
            printf("Point:(%f,%f),%s\n",this->X,this->Y,this->name.c_str());
        }
    }
    Line::Line():GeoObj(){
        this->X1=NONEXIST;
        this->X2=NONEXIST;
        this->Y1=NONEXIST;
        this->Y2=NONEXIST;
        this->line_type=line_type;
        this->type=GLINE;
    }
```

```
bool Line::update(void){
	if (this->determine_type==FREE){
		return false;
	}
	else if (this->determine_type==START_END){
		this->X1=((Point*)this->parent[0])->X;
		this->Y1=((Point*)this->parent[0])->Y;
		this->X2=((Point*)this->parent[1])->X;
		this->Y2=((Point*)this->parent[1])->Y;
	}
	ChildNode* current=this->children->start;
	while (current!=NULL){
		current->value->update();
	}
	return true;
}
double Line::length(){
	return sqrt((X1-X2)*(X1-X2)+(Y1-Y2)*(Y1-Y2));
}
Line* Line::LineXY(Point* P1,Point* P2,lin_type line_type){
	Sketch* sketch;
	if (P1->sketch!=P2->sketch) return NULL;
	else sketch=P1->sketch;
	Line* A=new Line();
```

```
        A->parent[0]=P1;
        A->parent[1]=P2;
        A->determine_type=START_END;
        A->line_type=line_type;
        P1->children->add(A);
        P2->children->add(A);
        A->update();
        return A;
}
double Line::getX1(){
        return this->X1;
}
double Line::getY1(){
        return this->Y1;
}
double Line::getX2(){
        return this->X2;
}
double Line::getY2(){
        return this->Y2;
}
Point* Line::getStart(){
        return (Point*)this->parent[0];
}
```

```
Point* Line::getEnd(){
	return (Point*)this->parent[1];
}
bool Line::setP1(double X1,double Y1){
	if(this->determine_type==FREE){
		this->X1=X1;
		this->Y1=Y1;
		this->update();
		return true;
	}
	else{
		return false;
	}
}
bool Line::setP2(double X2,double Y2){
	if(this->determine_type==FREE){
		this->X2=X2;
		this->Y2=Y2;
		this->update();
		return true;
	}
	else{
		return false;
	}
```

```
}
void Line::paint(){
	if (this->visible && !this->hasnon()){
		fprintf(_resultfile,"Line %f %f %f %f %d %s\n",this->X1,this->Y1,this->X2,this->Y2,this->line_type,this->name.c_str());
		printf("Line:(%f,%f)-(%f,%f),%s\n",this->X1,this->Y1,this->X2,this->Y2,this->name.c_str());
	}
}
int Line::getLineType(){
	if(this->line_type==SEGMENT)
		return 0;
	else if (this->line_type==RAY){
		return 1;
	}
	else {
		return 2;
	}
	return (int)this->line_type;
}
bool Line::hasnon(void){
	if (this->X1==NONEXIST||this->Y1==NONEXIST||this->X2==NONEXIST||this->Y2==NONEXIST){
		return true;
	}
```

```
        return false;
}


Circle::Circle():GeoObj(){
        this->X=NONEXIST;
        this->Y=NONEXIST;
        this->radius=NONEXIST;
        this->type=GCIRCLE;
}
Circle* Circle::DrawCircle(Point* center, Line* radius){
        if (center->sketch!=radius->sketch){
                return NULL;
        }
        Sketch* sketch=center->sketch;
        Circle* A=new Circle();
        A->determine_type=CENTER_RADIUS;
        A->parent[0]=center;
        A->parent[1]=radius;
        center->children->add(A);
        radius->children->add(A);
        A->update();
        return A;
}
```

```
Circle* Circle::DrawCircle(Point* center, double radius){
	Circle* A=new Circle();
	A->determine_type=CENTER;
	A->parent[0]=center;
	A->radius=radius;
	A->update();
	return A;
}
bool Circle::update(void){
	if (this->determine_type==FREE){
		return false;
	}else if(this->determine_type==CENTER){
		this->X=((Point*)this->parent[0])->X;
		this->Y=((Point*)this->parent[0])->Y;
	}else if(this->determine_type==CENTER_RADIUS){
		this->X=((Point*)this->parent[0])->X;
		this->Y=((Point*)this->parent[0])->Y;
		this->radius=((Line*)this->parent[1])->length();
	}
	ChildNode* current=this->children->start;
	while (current!=NULL){
		current->value->update();
	}
	return true;
```

```
}
Point* Circle::getCenter(){
	return (Point*)this->parent[0];
}
double Circle::getX(){
	return this->X;
}
double Circle::getY(){
	return this->Y;
}
double Circle::getRadius(){
	return this->radius;
}
bool Circle::setCenter(double X,double Y){
	if(this->determine_type==FREE){
		this->X=X;
		this->Y=Y;
		this->update();
		return true;
	}
	else{
		return false;
	}
}
```

```
bool Circle::setRadius(double r){
	if(this->determine_type==FREE||this->determine_type==CENTER){
		this->radius=r;
		this->update();
		return true;
	}
	return false;
}
void Circle::paint(){
	if (this->visible && !this->hasnon()){
		fprintf(_resultfile,"Circle %f %f %f %s\n",this->X,this->Y,this->radius,this->name.c_str());
		printf("Circle(%f,%f),%f,%s\n",this->X,this->Y,this->radius,this->name.c_str());
	}
}
bool Circle::hasnon(void){
	if (this->X==NONEXIST||this->Y==NONEXIST||this->radius==NONEXIST){
		return true;
	}
	return false;
}
```

double* GeoObj::intercept(GeoObj* obj1,GeoObj* obj2){//determine the intercept position of two geo-objects

```
	if ((obj1->type!=GCIRCLE&&obj1->type!=GLINE)||(obj1->type!=GCIRCLE&&obj1->type!=GLINE)){//not right type;
		return NULL;
	}
	if (obj1->type==GLINE&&obj2->type==GLINE){//both are lines;
		if (obj1->hasnon()||obj2->hasnon()){
			double* result=new double[2];
			result[0]=result[1]=NONEXIST;
			return  result;
		}
		Line* line1=(Line*)obj1;
		Line* line2=(Line*)obj2;
		line1->length();
		if (line1->length()<0.001*0.001){//the two points are too near, refuse to calculate intercept
			double* result=new double[2];
			result[0]=result[1]=NONEXIST;
			return  result;
		}
		else{
			double* result=new double[2];
			//calculate parameters in Ax+By+c=0 for line1 and line2
			double X1=line1->X1;
			double Y1=line1->Y1;
			double X2=line1->X2;
```

```
double Y2=line1->Y2;
double X3=line2->X1;
double Y3=line2->Y1;
double X4=line2->X2;
double Y4=line2->Y2;
double a1=Y1-Y2,b1=X2-X1,c1=X1*Y2-X2*Y1;
double a2=Y3-Y4,b2=X4-X3,c2=X3*Y4-X4*Y3;
if (abs(a1*b2-b1*a2)<0.001){//parallel
    result[0]=result[1]=NONEXIST;
    return result;
}
else{
    double X=(b2*c1-b1*c2)/(a2*b1-a1*b2);
    double Y=(a1*c2-a2*c1)/(a2*b1-a1*b2);
    bool is_in_line1=false;
    bool is_in_line2=false;
    if (line1->line_type==LINE || (line1->line_type==RAY &&
(X-X1)*(X2-X1)>=0 && (Y-Y1)*(Y2-Y1)>=0) || ((X-X1)*(X-X2)<=0 && (Y-Y1)*(Y-
Y2)<=0)){
        is_in_line1=true;
    }
    if (line2->line_type==LINE || (line2->line_type==RAY &&
(X-X3)*(X4-X3)>=0 && (Y-Y3)*(Y4-Y3)>=0) || ((X-X3)*(X-X4)<=0 && (Y-Y3)*(Y-
Y4)<=0)){
        is_in_line2=true;
    }
```

```
                    if (is_in_line1 && is_in_line2){
                        result[0]=X;
                        result[1]=Y;
                        return result;
                    }
                    else{
                        result[0]=result[1]=NONEXIST;
                        return result;
                    }
                }
            }
        }
        else if(obj1->type==GLINE||obj2->type==GLINE){//one of them is line, the other is circle.
            if (obj1->hasnon()||obj2->hasnon()){
                double* result=new double[4];
                result[0]=result[1]=result[2]=result[3]=NONEXIST;
                return  result;
            }
            double* result=new double[4];
            Line* line=(Line*)(obj1->type==GLINE?obj1:obj2);
            Circle* circle=(Circle*)(obj1->type==GLINE?obj2:obj1);
            double X1=line->X1,X2=line->X2,Y1=line->Y1,Y2=line->Y2;
            double X3=circle->X,Y3=circle->Y,r=circle->radius;
            double a1=Y1-Y2,b1=X2-X1,c1=X1*Y2-X2*Y1;
```

```
            if (abs(a1*X3+b1*Y3+c1)/sqrt(a1*a1+b1*b1)<=r+0.001){
        double Xnear=-(c1 + (b1*(a1*sqrt(a1*a1*r*r - a1*a1*X3*X3 -
2*a1*b1*X3*Y3 - 2*a1*c1*X3 + b1*b1*r*r - b1*b1*Y3*Y3 - 2*b1*c1*Y3 - c1*c1)+
a1*a1*Y3 - b1*c1 - a1*b1*X3))/(a1*a1 + b1*b1))/a1;
        double Xaway=-(c1 - (b1*(a1*sqrt(a1*a1*r*r - a1*a1*X3*X3 -
2*a1*b1*X3*Y3 - 2*a1*c1*X3 + b1*b1*r*r - b1*b1*Y3*Y3 - 2*b1*c1*Y3 - c1*c1)-
a1*a1*Y3 + b1*c1 + a1*b1*X3))/(a1*a1 + b1*b1))/a1;
        double Ynear= (a1*sqrt(a1*a1*r*r - a1*a1*X3*X3 - 2*a1*b1*X3*Y3 -
2*a1*c1*X3 + b1*b1*r*r - b1*b1*Y3*Y3 - 2*b1*c1*Y3 - c1*c1) + a1*a1*Y3 -
b1*c1 - a1*b1*X3)/(a1*a1 + b1*b1);
        double Yaway=-(a1*sqrt(a1*a1*r*r - a1*a1*X3*X3 - 2*a1*b1*X3*Y3 -
2*a1*c1*X3 + b1*b1*r*r - b1*b1*Y3*Y3 - 2*b1*c1*Y3 - c1*c1) - a1*a1*Y3 +
b1*c1 + a1*b1*X3)/(a1*a1 + b1*b1);
        result[0]=Xnear;
                    result[1]=Ynear;
                    result[2]=Xaway;
                    result[3]=Yaway;

            }
    else{

                    result[0]=result[1]=result[2]=result[3]=NONEXIST;

            }
            return result;
        }
        else{//two circles
```

```
        if (obj1->hasnon()||obj2->hasnon()){
            double* result=new double[4];
            result[0]=result[1]=result[2]=result[3]=NONEXIST;
            return  result;
        }
        double* result=new double[4];
        Circle* circle1=(Circle*)obj1;
        Circle* circle2=(Circle*)obj2;
        double x1=circle1->X,y1=circle1->Y,x2=circle2->X,y2=circle2->Y;
        double r1=circle1->radius,r2=circle2->radius;
        if ((x1-x2)*(x1-x2)+(y1-y2)*(y1-y2)>(r1+r2)*(r1+r2)||(x1-x2)*(x1-
x2)+(y1-y2)*(y1-y2)<(r1-r2)*(r1-r2)){
            result[0]=result[1]=result[2]=result[3]=NONEXIST;
    return result;
        }
        else{
            double Xnear,Ynear,Xaway,Yaway;
            if (abs(x1-x2)>0.1){
            Xnear=-(r1*r1 - r2*r2 - x1*x1 + x2*x2 - y1*y1 + y2*y2 +
(y1*(x2*sqrt((- r1*r1 + 2.*r1*r2 - r2*r2 + x1*x1 - 2.*x1*x2 + x2*x2 + y1*y1 -
2.*y1*y2 + y2*y2)*(r1*r1 + 2.*r1*r2 + r2*r2 - x1*x1 + 2.*x1*x2 - x2*x2 - y1*y1 +
2.*y1*y2 - y2*y2)) - x1*sqrt((- r1*r1 + 2.*r1*r2 - r2*r2 + x1*x1 - 2.*x1*x2 + x2*x2
+ y1*y1 - 2.*y1*y2 + y2*y2)*(r1*r1 + 2.*r1*r2 + r2*r2 - x1*x1 + 2.*x1*x2 - x2*x2 -
y1*y1 + 2.*y1*y2 - y2*y2)) - r1*r1*y1 + r1*r1*y2 + r2*r2*y1 - r2*r2*y2 +
x1*x1*y1 + x1*x1*y2 + x2*x2*y1 + x2*x2*y2 - y1*y2*y2 - y1*y1*y2 + y1*y1*y1 +
y2*y2*y2 - 2.*x1*x2*y1 - 2.*x1*x2*y2))/(x1*x1 - 2.*x1*x2 + x2*x2 + y1*y1 -
2.*y1*y2 + y2*y2) - (y2*(x2*sqrt((- r1*r1 + 2.*r1*r2 - r2*r2 + x1*x1 - 2.*x1*x2 +
```

x2*x2 + y1*y1 - 2.*y1*y2 + y2*y2)*(r1*r1 + 2.*r1*r2 + r2*r2 - x1*x1 + 2.*x1*x2 - x2*x2 - y1*y1 + 2.*y1*y2 - y2*y2)) - x1*sqrt((- r1*r1 + 2.*r1*r2 - r2*r2 + x1*x1 - 2.*x1*x2 + x2*x2 + y1*y1 - 2.*y1*y2 + y2*y2)*(r1*r1 + 2.*r1*r2 + r2*r2 - x1*x1 + 2.*x1*x2 - x2*x2 - y1*y1 + 2.*y1*y2 - y2*y2)) - r1*r1*y1 + r1*r1*y2 + r2*r2*y1 - r2*r2*y2 + x1*x1*y1 + x1*x1*y2 + x2*x2*y1 + x2*x2*y2 - y1*y2*y2 - y1*y1*y2 + y1*y1*y1 + y2*y2*y2 - 2.*x1*x2*y1 - 2.*x1*x2*y2))/(x1*x1 - 2.*x1*x2 + x2*x2 + y1*y1 - 2.*y1*y2 + y2*y2))/(2.*x1 - 2.*x2);

Xaway=-(r1*r1 - r2*r2 - x1*x1 + x2*x2 - y1*y1 + y2*y2 + (y1*(x1*sqrt((- r1*r1 + 2.*r1*r2 - r2*r2 + x1*x1 - 2.*x1*x2 + x2*x2 + y1*y1 - 2.*y1*y2 + y2*y2)*(r1*r1 + 2.*r1*r2 + r2*r2 - x1*x1 + 2.*x1*x2 - x2*x2 - y1*y1 + 2.*y1*y2 - y2*y2)) - x2*sqrt((- r1*r1 + 2.*r1*r2 - r2*r2 + x1*x1 - 2.*x1*x2 + x2*x2 + y1*y1 - 2.*y1*y2 + y2*y2)*(r1*r1 + 2.*r1*r2 + r2*r2 - x1*x1 + 2.*x1*x2 - x2*x2 - y1*y1 + 2.*y1*y2 - y2*y2))- r1*r1*y1 + r1*r1*y2 + r2*r2*y1 - r2*r2*y2 + x1*x1*y1 + x1*x1*y2 + x2*x2*y1 + x2*x2*y2 - y1*y2*y2 - y1*y1*y2 + y1*y1*y1 + y2*y2*y2 - 2.*x1*x2*y1 - 2.*x1*x2*y2))/(x1*x1 - 2.*x1*x2 + x2*x2 + y1*y1 - 2.*y1*y2 + y2*y2) - (y2*(x1*sqrt((- r1*r1 + 2.*r1*r2 - r2*r2 + x1*x1 - 2.*x1*x2 + x2*x2 + y1*y1 - 2.*y1*y2 + y2*y2)*(r1*r1 + 2.*r1*r2 + r2*r2 - x1*x1 + 2.*x1*x2 - x2*x2 - y1*y1 + 2.*y1*y2 - y2*y2)) - x2*sqrt((- r1*r1 + 2.*r1*r2 - r2*r2 + x1*x1 - 2.*x1*x2 + x2*x2 + y1*y1 - 2.*y1*y2 + y2*y2)*(r1*r1 + 2.*r1*r2 + r2*r2 - x1*x1 + 2.*x1*x2 - x2*x2 - y1*y1 + 2.*y1*y2 - y2*y2)) - r1*r1*y1 + r1*r1*y2 + r2*r2*y1 - r2*r2*y2 + x1*x1*y1 + x1*x1*y2 + x2*x2*y1 + x2*x2*y2 - y1*y2*y2 - y1*y1*y2 + y1*y1*y1 + y2*y2*y2 - 2.*x1*x2*y1 - 2.*x1*x2*y2))/(x1*x1 - 2.*x1*x2 + x2*x2 + y1*y1 - 2.*y1*y2 + y2*y2))/(2.*x1 - 2.*x2);

Ynear=(x2*sqrt((- r1*r1 + 2.*r1*r2 - r2*r2 + x1*x1 - 2.*x1*x2 + x2*x2 + y1*y1 - 2.*y1*y2 + y2*y2)*(r1*r1 + 2.*r1*r2 + r2*r2 - x1*x1 + 2.*x1*x2 - x2*x2 - y1*y1 + 2.*y1*y2 - y2*y2)) - x1*sqrt((- r1*r1 + 2.*r1*r2 - r2*r2 + x1*x1 - 2.*x1*x2 + x2*x2 + y1*y1 - 2.*y1*y2 + y2*y2)*(r1*r1 + 2.*r1*r2 + r2*r2 - x1*x1 + 2.*x1*x2 - x2*x2 - y1*y1 + 2.*y1*y2 - y2*y2)) - r1*r1*y1 + r1*r1*y2 + r2*r2*y1 - r2*r2*y2 + x1*x1*y1 + x1*x1*y2 + x2*x2*y1 + x2*x2*y2 - y1*y2*y2 - y1*y1*y2 + y1*y1*y1 + y2*y2*y2 - 2.*x1*x2*y1 - 2.*x1*x2*y2)/(2*(x1*x1 - 2.*x1*x2 + x2*x2 + y1*y1 - 2.*y1*y2 + y2*y2));

Yaway=(x1*sqrt((- r1*r1 + 2.*r1*r2 - r2*r2 + x1*x1 - 2.*x1*x2 + x2*x2 + y1*y1 - 2.*y1*y2 + y2*y2)*(r1*r1 + 2.*r1*r2 + r2*r2 - x1*x1 + 2.*x1*x2 - x2*x2 -

```
y1*y1 + 2.*y1*y2 - y2*y2)) - x2*sqrt((- r1*r1 + 2.*r1*r2 - r2*r2 + x1*x1 - 2.*x1*x2
+ x2*x2 + y1*y1 - 2.*y1*y2 + y2*y2)*(r1*r1 + 2.*r1*r2 + r2*r2 - x1*x1 + 2.*x1*x2 -
x2*x2 - y1*y1 + 2.*y1*y2 - y2*y2)) - r1*r1*y1 + r1*r1*y2 + r2*r2*y1 - r2*r2*y2 +
x1*x1*y1 + x1*x1*y2 + x2*x2*y1 + x2*x2*y2 - y1*y2*y2 - y1*y1*y2 + y1*y1*y1 +
y2*y2*y2 - 2.*x1*x2*y1 - 2.*x1*x2*y2)/(2*(x1*x1 - 2.*x1*x2 + x2*x2 + y1*y1 -
2.*y1*y2 + y2*y2));
                        }
                        else{
                                double x0=x1;


                                Xnear=(2*x0*y1 - 2*x0*y2 + sqrt(-(r1 + r2 + y1 - y2)*(r1
+ r2 - y1 + y2)*(r1 - r2 + y1 - y2)*(r1 - r2 - y1 + y2)))/(2*(y1 - y2));
                                Xaway=-(2*x0*y2 - 2*x0*y1 + sqrt(-(r1 + r2 + y1 - y2)*(r1
+ r2 - y1 + y2)*(r1 - r2 + y1 - y2)*(r1 - r2 - y1 + y2)))/(2*(y1 - y2));
                                Ynear=-(r1*r1 - r2*r2 - y1*y1 + y2*y2)/(2*(y1 - y2));
                                Yaway=Ynear;
                        }
                        result[0]=Xnear;
                        result[1]=Ynear;
                        result[2]=Xaway;
                        result[3]=Yaway;
                        return result;

                }

        }
```

```
}
void Sketch::paint(){
	Node* current=eleList->start;
	while (current!=NULL){
		current->value->paint();
		current=current->next;
	}
}
//*********************************************************
	variable::variable(){
		this->next=NULL;
	}
	bool Symbol_Table_Node::operator!=(const Symbol_Table_Node
&orig)const{
		return !(this == &orig);
	}
	Symbol_Table_Node::Symbol_Table_Node(Symbol_Table_Node* parent){
		this->First=this->Last=NULL;
```

```
        this->Firstf=this->Lastf=0;
        this->parent=parent;
    }
    Symbol_Table_Node::~Symbol_Table_Node(){
        for (variable* current=this->First;current!=NULL;current=current-
>next){
            switch (current->var_type){
            case var_float:
                delete current->var_strorage.val_float;
                break;
            case var_num:
                delete current->var_strorage.val_int;
                break;
            case var_string:
                delete current->var_strorage.val_string;
                break;
            case var_bool:
                delete current->var_strorage.val_bool;
                break;
            default:
                ;
            }
        }
    }
    Symbol_Table_Node* Symbol_Table_Node::add_node(){
```

```
        Symbol_Table_Node* new_node=new Symbol_Table_Node(this);
        return new_node;
}
variable* Symbol_Table_Node::add_var(int value,string name){
        int* a=new int(value);
        variable* new_var=new variable;
        new_var->var_type=var_num;
        new_var->var_strorage.val_int=a;
        new_var->name=name;
        if (this->Last==NULL){
                this->First=this->Last=new_var;
                return new_var;
        }
        this->Last->next=new_var;
        this->Last=new_var;
        return new_var;
}
variable* Symbol_Table_Node::add_var(string value,string name){
        string* a=new string(value);
        variable* new_var=new variable;
        new_var->var_type=var_string;
        new_var->var_strorage.val_string=a;
        new_var->name=name;
        if (this->Last==NULL){
```

```
            this->First=this->Last=new_var;
            return new_var;
        }
        this->Last->next=new_var;
        this->Last=new_var;

        return new_var;
}
variable* Symbol_Table_Node::add_var(double value,string name){
        variable* new_var=new variable;
        double* a=new double(value);
        new_var->var_type=var_float;
        new_var->var_strorage.val_float=a;
        new_var->name=name;
        if (this->Last==NULL){
            this->First=this->Last=new_var;
            return new_var;
        }
        this->Last->next=new_var;
        this->Last=new_var;
        return new_var;
}
variable* Symbol_Table_Node::add_var(bool value,string name){
        bool* a=new bool(value);
```

```
        variable* new_var=new variable;
        new_var->var_type=var_bool;
        new_var->var_strorage.val_bool=a;
        new_var->name=name;
        if (this->Last==NULL){
                this->First=this->Last=new_var;
                return new_var;
        }
        this->Last->next=new_var;
        this->Last=new_var;
        return new_var;
}
variable* Symbol_Table_Node::add_var(Point* value,string name){
        variable* new_var=new variable;
        new_var->var_type=var_point;
        new_var->var_strorage.val_point=value;
        new_var->name=name;
        if (this->Last==NULL){
                this->First=this->Last=new_var;
                return new_var;
        }
        this->Last->next=new_var;
        this->Last=new_var;
        return new_var;
```

```
}
variable* Symbol_Table_Node::add_var(Circle* value,string name){
	variable* new_var=new variable;
	new_var->var_type=var_circle;
	new_var->var_strorage.val_circle=value;
	new_var->name=name;
	if (this->Last==NULL){
		this->First=this->Last=new_var;
		return new_var;
	}
	this->Last->next=new_var;
	this->Last=new_var;

	return new_var;
}
variable* Symbol_Table_Node::add_var(Line* value,string name){
	variable* new_var=new variable;
	new_var->var_type=var_line;
	new_var->var_strorage.val_line=value;
	new_var->name=name;
	if (this->Last==NULL){
		this->First=this->Last=new_var;
		return new_var;
	}
```

```
        this->Last->next=new_var;
        this->Last=new_var;

        return new_var;
}
void Symbol_Table_Node::add_function(func_pointer funcp,string name){
        function* new_func= new function;
        new_func->funcp=funcp;
        new_func->name=name;
        if (this->Lastf==NULL){
                this->Firstf=this->Lastf=new_func;
                return;
        }
        this->Lastf->next=new_func;
        this->Lastf=new_func;
}
void* Symbol_Table_Node::get_var(string name,bool is_local){
        variable* current=this->First;
        while (current!=NULL){
                if (current->name==name){
                        switch (current->var_type){
                        case var_float:
                                return current->var_strorage.val_float;
                        case var_num:
```

```
                return current->var_strorage.val_int;
            case var_string:
                return current->var_strorage.val_string;
            case var_bool:
                return current->var_strorage.val_bool;
            case var_point:
                return &(current->var_strorage.val_point);
            case var_circle:
                return &(current->var_strorage.val_circle);
            case var_line:
                return &(current->var_strorage.val_line);
            default:
                ;
            }
        }else{
            current=current->next;
        }

}
if (is_local){
        return NULL;
        }
        else if (this->parent==NULL){
            return NULL;
```

```
                }
                else{
                        return this->parent->get_var(name,is_local);
                }
}
func_pointer Symbol_Table_Node::get_function(string name,bool is_local){
        function* current=this->Firstf;
        while (current!=NULL){
                if (current->name==name){
                        return current->funcp;
                }else{
                        current=current->next;
                }
        }
        if (is_local){
                return NULL;
                }
                else if (this->parent==NULL){
                        return NULL;
                }
                else{
                        return this->parent->get_function(name,is_local);
                }
}
```

```
int lib::print(string a){
	cout<<a;
	return 0;
}
double lib::sin(double x){
	return sqrt(x);
}
Line* lib::LineST(Point* P1,Point* P2,int line_type){
	//lin_type line_type=SEGMENT
	return Line::LineXY(P1,P2,(lin_type) line_type);
}
Circle* lib::DrawCircle(Point* center, Line* radius){
	return Circle::DrawCircle(center,radius);
}
Circle* lib::DrawCircle(Point* center, double radius){
	return Circle::DrawCircle(center,radius);
}
Point* lib::intersect(GeoObj* GeoObj1,GeoObj* GeoObj2,bool isnear){
	Point* a=Point::intersect(GeoObj1,GeoObj2,isnear);
```

```
        return a;
}
Point* lib::PointXY(double X,double Y){
        Point* a= new Point;
        a->setXY(X,Y);
        return a;
}
Line* lib::LineXY(double X1,double Y1,double X2,double Y2,int line_type){
        Point* o1=this->PointXY(X1,Y1);
        Point* o2=this->PointXY(X2,Y2);
        Line* a=this->LineST(o1,o2,(lin_type) line_type);
        return a;
}
Circle* lib::CircleXYr(double X,double Y,double r){
        Point* o=this->PointXY(X,Y);
        Circle* a=this->DrawCircle(o,r);
        a->setCenter(X,Y);
        a->setRadius(r);
        return a;
}
void lib::paint(){
        fprintf(_resultfile,"START\n");
        _sketch->paint();
        fprintf(_resultfile,"END\n");
```

```
        }
```

2. DocForm.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace 多文档编辑器
{
   public partial class DocForm : Form
   {
      DocForm doc;

      public RichTextBox Sourse
      {
         get
         {
            return richTextBox;
```

```
    }
    set
    {
        richTextBox = value;
    }
}


private string _filePath = string.Empty;
private int _index;

public int GetFileTypeIndex()
{
    return _index;
}
public void SetFileTypeIndex(int i)
{
    _index = i;
}


public string GetFilePath()
{
    return this._filePath;
```

```
}
public void SetFilePath(string value)
{
   _filePath = value;
}


public DocForm()
{
   InitializeComponent();
}
public DocForm(RichTextBoxStreamType fileType, string filePath,int i)
{
   InitializeComponent();

   this.SetFilePath(filePath);
   this.richTextBox.LoadFile(filePath, fileType);
   this.SetFileTypeIndex(i);
}


private void DocForm_Load(object sender, EventArgs e)
{
   this.doc = (DocForm)MainForm.GetDocTrun();
```

```
        }
    }
}
```

3. DrawForm.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Media;

namespace 多文档编辑器
{
    public partial class DrawForm : Form
    {
        private System.Drawing.Graphics graphics;
        public DrawForm()
        {
            InitializeComponent();
```

```
   graphics = this.CreateGraphics();


}
~DrawForm()
{




}
public void drawLine(float x1, float y1, float x2, float y2, int label)
{

  // x1 = x1  + 500; y1 = y1  + 500; x2 = x2  + 500; y2 = y2  + 500;
       Brush br = new SolidBrush(Color.Black);
       Pen p = new Pen(br);
       graphics.DrawLine(p,x1,y1,x2,y2);
      // graphics.DrawRectangle(p,10,10,5,5);
       if (label == 1)
       {//segment
          drawPoint(x1, y1);
          drawPoint(x2, y2);
       }
       else if (label == 3)
       {//ray
```

```
            drawPoint(x1, y1);
        }
        else if (label == 2)
        {
            ;
        }
        else
        {

        }

 }

public void drawCircle(float x, float y, float radius, int label )
{
  //  x = x ; y = y ; radius = radius +200;

    Brush br = new SolidBrush(Color.Black);
    Pen p = new Pen(br);
    if (label == 1)
    {
        graphics.DrawEllipse(p, x, y, radius, radius);
    }
    else
```

```
        {
            graphics.DrawEllipse(p, x, y, radius, radius);
            graphics.FillEllipse(br, x, y, radius, radius);
        }

    }

    public void drawPoint(float x, float y)
    {
        drawCircle(x,y, 3, 2);

    }


    }

}
```

4. FindForm.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
```

```
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace 多文档编辑器
{
    public partial class FindForm : Form
    {
        DocForm doc ;
        private static int index=0;
        private static int count = 0;
        public FindForm()
        {
            InitializeComponent();
        }

        private void FindForm_Load(object sender, EventArgs e)
        {
            this.doc = MainForm.GetDocTrun();
        }
```

```
private void btn_FindFrom_ReplaceOne_Click(object sender, EventArgs e)
{
   string strOld = Convert.ToString(tb_FindForm_FinfContain.Text);
   string strNew = Convert.ToString(tb_FindForm_Replace.Text);
   string str = Convert.ToString(tb_FindForm_FinfContain.Text);

   try
   {
      if (doc.Sourse.Text.Contains(strOld))
      {
         Clipboard.Clear();
         count = doc.Sourse.Find(strOld, index, RichTextBoxFinds.None);
         doc.Sourse.Select(count, str.Length);
         Clipboard.SetText(doc.Sourse.SelectedText.Replace(strOld, strNew));
         doc.Sourse.Paste();
         count++;
      }
      else
         return;
   }
   catch (Exception ex)
   {
      return;
```

```
    }
}

private void btn_FindFrom_ReplaceAll_Click(object sender, EventArgs e)
{
    string strOld = Convert.ToString(tb_FindForm_FinfContain.Text);
    string strNew = Convert.ToString(tb_FindForm_Replace.Text);

    try
    {
        if (doc.Sourse.Text.Contains(strOld))
        {
            doc.Sourse.Text = doc.Sourse.Text.Replace(strOld, strNew);
        }
    }
    catch (Exception ex)
    {
        return;
    }
}

private void btn_FindFrom_FindNext_Click(object sender, EventArgs e)
{
    string str = Convert.ToString(tb_FindForm_FinfContain.Text);
```

```
    try
    {
        if (doc.Sourse.Text.Contains(str))
        {
            index = doc.Sourse.Find(str, index, RichTextBoxFinds.None);
            doc.Sourse.Select(index, str.Length);
            index++;
        }
    }
    catch (Exception ex)
    {
        if (index == -1)
        {
            index = 0;
        }
    }
}

private void btn_FindFrom_Cancel_Click(object sender, EventArgs e)
{
    this.Close();
}

private void label1_Click(object sender, EventArgs e)
```

```
        {


        }
    }
}
```

5. MainForm.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.OleDb;
using System.Drawing;
using System.Drawing.Drawing2D;
using System.Drawing.Printing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.IO;
using System.Reflection;
using System.Runtime.InteropServices;
using System.Diagnostics;
```

```
using System.Drawing;
namespace 多文档编辑器
{
   public partial class MainForm : Form
   {
      private int wCount = 0;
      private RichTextBoxStreamType oldFileType;
      private DocForm doc;
      private DrawForm drawform;

      private static DocForm docSend;
      private string fileName = string.Empty;
      private string filePath = string.Empty;

      List<DocForm> listDocForm = new List<DocForm>();

      string _connectionString =
string.Format(@"Provider=Microsoft.Jet.OLEDB.4.0;Data
Source={0}\FileData.mdb", Environment.CurrentDirectory);

      public MainForm()
      {
         InitializeComponent();
      }
```

```
    public static DocForm GetDocTrun()
    {
       return docSend;
    }



    private void tsmi_Format_Wrap_Click(object sender, EventArgs e)    //自动换行
    {
       try
       {
          if (tsmi_Format_Wrap.CheckState == CheckState.Checked)
          {
             tssl_FormCount.Text = string.Format("   Successfully disable format wrap function ......");
             tsmi_Format_Wrap.CheckState = CheckState.Unchecked;
             DocForm df = (DocForm)this.ActiveMdiChild;
             df.Sourse.WordWrap = false;
          }
          else
          {
             tssl_FormCount.Text = string.Format("   Successfully enable format wrap function ......");
             tsmi_Format_Wrap.CheckState = CheckState.Checked;
```

```
            DocForm df = (DocForm)this.ActiveMdiChild;
            df.Sourse.WordWrap = true;
        }
    }
    catch (Exception ex)
    {
        return;
    }
}

private void tsmi_Format_Font_Click(object sender, EventArgs e)    //字体大小设置
{
    tssl_FormCount.Text = string.Format("   Enable word size setting ......");
    try
    {
        if (fontDialog1.ShowDialog() == DialogResult.OK && doc != null)
        {
            DocForm df = (DocForm)this.ActiveMdiChild;
            df.Sourse.SelectionFont = fontDialog1.Font;
        }
    }
    catch (Exception ex)
    {
        return;
```

```
    }
    tssl_FormCount.Text = string.Format("   Disable word size setting ......");
}

private void tsmi_Format_Colour_Click(object sender, EventArgs e)   // 字体颜色设置
{
    tssl_FormCount.Text = string.Format("   Enable word color setting ......");
    try
    {
        if (colorDialog1.ShowDialog() == DialogResult.OK && doc != null)
        {
            DocForm df = (DocForm)this.ActiveMdiChild;
            df.Sourse.SelectionColor = colorDialog1.Color;
        }
    }
    catch (Exception ex)
    {
        return;
    }
}
```

```
    private void tsmi_File_NewCreate_Click(object sender, EventArgs e)  //新建文件
    {
        wCount++;
        doc = new DocForm();
        listDocForm.Add(doc);

        doc.MdiParent = this;
        doc.Text = "文档" + wCount;
        doc.Show();

        if (wCount == 1)
        {
            doc.WindowState = FormWindowState.Maximized;
        }
        tssl_FormCount.Text = string.Format("   Successfully create file ......Current filename is ", this.MdiChildren.Length, doc.Text);
    }

    private void tsmi_File_Open_Click(object sender, EventArgs e)      //打开文件
    {
        tssl_FormCount.Text = string.Format("  Open file ......");
        doc = new DocForm();
        try
        {
```

```
            if (openFileDialog1.ShowDialog() == DialogResult.OK)
            {
               RichTextBoxStreamType fileType =
TrunFileType(openFileDialog1.FilterIndex);

            //    MessageBox.Show("time after fileType");
            //    MessageBox.Show("file name is "+openFileDialog1.FileName);
               wCount++;
               doc = new DocForm(fileType, openFileDialog1.FileName,
openFileDialog1.FilterIndex);
               doc.MdiParent = this;
               doc.Show();
               listDocForm.Add(doc);
             //   MessageBox.Show("time after listdocform");
            }
         }
         catch (Exception ex)
         {
            return;
         }
        //  MessageBox.Show("time to str");
         string str = openFileDialog1.FileName;
         string[] sArray = str.Split('\\');
         if (sArray.Length == 0) return;
```

```
            doc.Text = sArray[sArray.Length - 1];


            tssl_FormCount.Text = string.Format("   Successfully open file, the filename
is  ", doc.Text);
            doc.WindowState = FormWindowState.Maximized;
        }


        private void tsmi_File_Save_Click(object sender, EventArgs e)      //保存文件
        {
            DocForm df = (DocForm)this.ActiveMdiChild;
            try
            {
                if (df.GetFilePath() == "")
                {
                    tssl_FormCount.Text = string.Format("   Please select save path ");
                    SaveFile(df);
                    //SaveDateToDateBase(df);
                    MessageBox.Show("save successfully");
                    tssl_FormCount.Text = string.Format("   save successfully ", doc.Text);
                }
                else
                {
                    RichTextBoxStreamType fileType = TrunFileType(df.GetFileTypeIndex());
                    df.Sourse.SaveFile(df.GetFilePath(), fileType);
                    MessageBox.Show("save successfully");
```

```
            tssl_FormCount.Text = string.Format("   save successfully ", doc.Text);
        }

        //save the path to log
        saveLog(df.GetFilePath());



    }
    catch (Exception ex)
    {
        MessageBox.Show("Failure, save again !  ");
    }
}

private void saveLog(string savepath)
{
    String path = System.Environment.CurrentDirectory + "\\log.txt";
    StreamReader rd = new StreamReader(path);
    string st = string.Empty;
    string[] content = { null, null, null };
    //if the path exists in the loop
    int count = 0;
    int exist = 0;
```

```
while (!rd.EndOfStream && count < 2)
{
   st = rd.ReadLine();



   if (st == savepath)
   { exist = 1; break; }
   ++count;
   content[count] = st;
   // MessageBox.Show("count is " + (count - 1) + " and " + content[count]);
}
rd.Close();

//if the path is a new path
if (exist == 0)
{
   content[0] = savepath;
   int i = 0;
   StreamWriter sw = File.CreateText(path);
   sw.Write("");
   sw.Close();
   //   MessageBox.Show("there is " + (count + 1) + " and the content0: " +
content[0] + " and content1: " + content[1] + " and content2" + content[2]);
```

```
            StreamWriter sww = new StreamWriter(path);
            while (i < 3 && content[i] != null)
            {
                //      MessageBox.Show("in the danger loop to show content[" + i +
" ] :  "+content[i]);
                sww.WriteLine(content[i]);
                //      MessageBox.
                //          Show("danger end.");
                i++;
            }

            sww.Close();
            //  MessageBox.
            //      Show("exist end.");
        }
        //  MessageBox.Show("check function end");
    }

    private void tsmi_File_OtherSave_Click(object sender, EventArgs e)  //另存为
    {
        tssl_FormCount.Text = string.Format("   Please select save path ");
        DocForm df = (DocForm)this.ActiveMdiChild;
        try
        {
            SaveFile(df);
```

```
            filePath = df.GetFilePath();
            //SaveDateToDateBase(df);
            MessageBox.Show("save successfully");
        }
        catch (Exception ex)
        {
            return;
        }
        tssl_FormCount.Text = string.Format("   Save successfully ", doc.Text);
        saveLog(df.GetFilePath());

    }

    /*    private void tsmi_File_SaveToDateBase_Click(object sender, EventArgs
e)//保存到数据库
        {
            try
            {
                DocForm df = (DocForm)this.ActiveMdiChild;
                fileName = df.Text;
                if (filePath == "")
                {
                    filePath = Environment.CommandLine;
                }
                SaveDateToDateBase(df);
```

```
        }
        catch (Exception ex)
        {
           return;
        }
      }*/

    private void tsmi_File_PrintPage_Click(object sender, EventArgs e)  //打印设置
    {
       tssl_FormCount.Text = string.Format("   Setting printer..... ");
       PrintDialog pd = new PrintDialog();
       pd.ShowDialog();
    }

    private void tsmi_File_PageSet_Click(object sender, EventArgs e)    //页面设置
    {
       tssl_FormCount.Text = string.Format("   Setting page..... ");
       pageSetupDialog1.Document = new PrintDocument();

       this.pageSetupDialog1.AllowMargins = true;
       this.pageSetupDialog1.AllowOrientation = true;
       this.pageSetupDialog1.AllowPaper = true;
       this.pageSetupDialog1.AllowPrinter = true;
```

```
    this.pageSetupDialog1.ShowDialog();
}

private void tsmi_File_Exit_Click(object sender, EventArgs e)      //退出程序
{
    Application.Exit();
}



private void tsmi_Edit_Copy_Click(object sender, EventArgs e)   //复制
{
    DocForm df = (DocForm)this.ActiveMdiChild;
    try
    {
        if (df.Sourse.SelectionLength > 0)
        {
            df.Sourse.Copy();
        }
    }
    catch
    {
        return;
```

```
    }
    tssl_FormCount.Text = string.Format("   Copy successfully   ");
}

private void tsmi_Edit_Stick_Click(object sender, EventArgs e)  //粘贴
{
    DocForm df = (DocForm)this.ActiveMdiChild;
    try
    {
        df.Sourse.Paste();
        tssl_FormCount.Text = string.Format("   Paste successfully   ");
    }
    catch (Exception ex)
    {
        return;
    }
}

private void tsmi_Edit_Cut_Click(object sender, EventArgs e)    //剪接
{
    DocForm df = (DocForm)this.ActiveMdiChild;
    try
    {
        if (df.Sourse.SelectionLength > 0)
```

```
        {
            df.Sourse.Cut();
        }
    }
    catch (Exception ex)
    {
        return;
    }
    tssl_FormCount.Text = string.Format("   Cut successfully   ");
}

private void tsmi_Edit_Delete_Click(object sender, EventArgs e) //删除
{
    DocForm df = (DocForm)this.ActiveMdiChild;
    try
    {
        if (df.Sourse.SelectionLength > 0)
        {
            df.Sourse.SelectedText = "";
        }
    }
    catch (Exception ex)
    {
        return;
```

```
    }
    tssl_FormCount.Text = string.Format("   Delete successfully   ");
}

private void tsmi_Edit_Cancel_Click(object sender, EventArgs e) //撤销
{
    DocForm df = (DocForm)this.ActiveMdiChild;
    try
    {
        if (df.Sourse.CanUndo == true)
        {
            df.Sourse.Undo();
        }
    }
    catch (Exception ex)
    {
        return;
    }
    tssl_FormCount.Text = string.Format("   Undo successfully   ");
}

private void tsmi_Edit_Find_Click(object sender, EventArgs e)   //查找
{
    docSend = (DocForm)this.ActiveMdiChild;
```

```
    FindForm find = new FindForm();
    find.Show();
    tssl_FormCount.Text = string.Format("   Find for Editing   ");
}



private void tsmi_Edit_AllSelect_Click(object sender, EventArgs e)  //全选
{
    try
    {
        DocForm df = (DocForm)this.ActiveMdiChild;
        df.Sourse.SelectAll();
        tssl_FormCount.Text = string.Format("   Already select all.....   ");
    }
    catch (Exception ex)
    {
        return;
    }
}

private void tsmi_Edit_Date_Click(object sender, EventArgs e)   //日期时间
{
    try
```

```
        {
            DocForm df = (DocForm)this.ActiveMdiChild;
            Clipboard.SetText(DateTime.Now.ToString());
            df.Sourse.Paste();
            tssl_FormCount.Text = string.Format("   Edit date.....   ");
        }
        catch (Exception ex)
        {
            return;
        }
    }

    private void tsmi_Help_LookForHelp_Click(object sender, EventArgs e)    //帮助信息
    {
        tssl_FormCount.Text = string.Format("    Open help.....");
        MessageBox.Show("Thanks for using, please contact the developer. ");
    }

    private void tsmi_Help_AboutTheSoft_Click(object sender, EventArgs e)   //关于软件信息
    {
        tssl_FormCount.Text = string.Format("    About this software.....");
```

```
        MessageBox.Show("Software version： 1.0\n\rDevelopers: Yan, Zhongyu, Yichen\n\rContact： xxxx@hotmail.com");
    }

    private void tsmi_Check_Statue_Click(object sender, EventArgs e)      //状态栏
    {
      try
      {
        if (tsmi_Check_Statue.CheckState == CheckState.Checked)
        {

          tsmi_Check_Statue.CheckState = CheckState.Unchecked;
          stasp_StatueBar.Visible = false;
        }
        else
        {
          tsmi_Check_Statue.CheckState = CheckState.Checked;
          stasp_StatueBar.Visible = true;
          tssl_FormCount.Text = string.Format("   Enable status bar ");
        }
      }
      catch (Exception ex)
      {
        return;
```

```
    }
}
```

```
public void ExecuteSql(string sql)              //执行 sql 语句
{
    using (OleDbConnection conn = new OleDbConnection(_connectionString))
    {
        using (OleDbCommand command = new OleDbCommand())
        {
            try
            {
                conn.Open();
                command.Connection = conn;
                command.CommandType = System.Data.CommandType.Text;
                command.CommandText = sql;
                command.ExecuteNonQuery();
            }
            catch (Exception ex)
            {
                conn.Close();
            }
```

```
        }
    }
}

public DataTable ReadDataToDataTable(string sql)   //输出数据库数据
{
    using (OleDbConnection conn = new OleDbConnection(_connectionString))
    {
        using (OleDbCommand command = new OleDbCommand())
        {

            conn.Open();
            command.Connection = conn;
            command.CommandType = System.Data.CommandType.Text;

            DataTable dt = new DataTable();
            OleDbDataAdapter adapter = new OleDbDataAdapter();

            command.CommandText = sql;
            adapter.SelectCommand = command;
            adapter.Fill(dt);
            return dt;
        }
    }
```

```
    }

    public void SaveFile(DocForm df)                //保存文件
    {
        SaveFileDialog sfd = new SaveFileDialog();
        sfd.Filter = "文本文件（*.txt）|*.txt|RTF 文件|*.rtf|所有文件（*.*）
|*.*";

        if (sfd.ShowDialog() == DialogResult.OK)
        {
            RichTextBoxStreamType fileType = TrunFileType(sfd.FilterIndex);
            doc.SetFileTypeIndex(sfd.FilterIndex);
            doc.SetFilePath(saveFileDialog1.InitialDirectory);
            df.Sourse.SaveFile(sfd.FileName, fileType);

            df.SetFilePath(sfd.FileName);
            oldFileType = fileType;
        }
        string str = df.GetFilePath();

        string[] sArray = str.Split('\\');
        fileName = sArray[sArray.Length - 1];

        for (int i = 0; i < sArray.Length - 1; i++)
        {
```

```
            filePath += sArray[i];
        }
    }

    public void SaveDateToDateBase(DocForm df)        //保存到数据库
    {
        try
        {
            DataTable dtPath = new DataTable();
            dtPath = ReadDataToDataTable("select fNumber,fName from tFilePath");

            DataTable dtContent = new DataTable();
            dtContent = ReadDataToDataTable("select fNumber,fName from
tFileContent");

            if (dtPath.Rows.Count != 0)
            {
                for (int i = 0; i < dtPath.Rows.Count; i++)
                {
                    if (fileName == (Convert.ToString(dtPath.Rows[i]["fName"])))
                    {
                        int idPath = Convert.ToInt32(dtPath.Rows[i]["fNumber"]);
                        ExecuteSql(string.Format("update tFilePath set fPath='{0}', where
fNumber='{1}'", filePath, idPath));
                    }
```

```
                else
                    ExecuteSql(string.Format("insert into tFilePath(fName,fPath)
values('{0}','{1}')", fileName, filePath));
                }
            }
            else
                ExecuteSql(string.Format("insert into tFilePath(fName,fPath)
values('{0}','{1}')", fileName, filePath));



            if (dtContent.Rows.Count != 0)
            {
                for (int i = 0; i < dtContent.Rows.Count; i++)
                {
                    if (fileName == (Convert.ToString(dtContent.Rows[i]["fName"])))
                    {
                        int idContent = Convert.ToInt32(dtContent.Rows[i]["fNumber"]);
                        ExecuteSql(string.Format("update tFileContent set
fContent='{0}',fDate='{1}', where fNumber='{2}'", df.Sourse.Text, DateTime.Now,
idContent));
                    }
                    else
                        ExecuteSql(string.Format("insert into
tFileContent(fName,fContent,fDate) values('{0}','{1}','{2}')", fileName,
df.Sourse.Text, DateTime.Now));
```

```
            }
        }
        else
            ExecuteSql(string.Format("insert into
tFileContent(fName,fContent,fDate) values('{0}','{1}','{2}')", fileName,
df.Sourse.Text, DateTime.Now));
        MessageBox.Show("Successfully save to database！");
    }
    catch (Exception ex)
    {
        MessageBox.Show("Save failure！");
    }
}

public RichTextBoxStreamType TrunFileType(int i)  //数据转换
{
    RichTextBoxStreamType fileType;
    switch (i)
    {
        case 1: fileType = RichTextBoxStreamType.PlainText;
            break;
        case 2: fileType = RichTextBoxStreamType.RichText;
            break;
        default: fileType = RichTextBoxStreamType.UnicodePlainText;
            break;
```

```
        }
        return fileType;
    }

    private void tsmi_File_History_Click(object sender, EventArgs e)
    {

        addItems();



    }
    private void item1_Click(object sender, EventArgs e)
    {
        //MessageBox.Show("aha");
        string path =
(string)this.item3.GetType().GetProperty("Text").GetValue(this.item1);

        showText(path);

    }

    private void addItems()
    {
        ToolStripMenuItem o = new ToolStripMenuItem();
```

```
        StreamReader rd = new
StreamReader(System.Environment.CurrentDirectory + "\\log.txt");
        if (!rd.EndOfStream)
        {
            this.item1.GetType().GetProperty("Visible").SetValue(this.item1, true);
            this.item1.GetType().GetProperty("Text").SetValue(this.item1,
rd.ReadLine());
        }
        if (!rd.EndOfStream)
        {
            this.item2.GetType().GetProperty("Visible").SetValue(this.item2, true);
            this.item2.GetType().GetProperty("Text").SetValue(this.item2,
rd.ReadLine());
        }
        if (!rd.EndOfStream)
        {
            this.item3.GetType().GetProperty("Visible").SetValue(this.item3, true);
            this.item3.GetType().GetProperty("Text").SetValue(this.item3,
rd.ReadLine());
        }
        rd.Close();
    }

    private void menuStrip1_ItemClicked(object sender,
ToolStripItemClickedEventArgs e)
    {
```

```
}

private void textBox1_TextChanged(object sender, EventArgs e)
{

}
[DllImport("kernel32.dll")]
public static extern int WinExec(string exeName, int operType);

private void copyFile(string sourcepath, string destpath)
{

   while (!System.IO.File.Exists(destpath))
      System.IO.File.Create(destpath);
   // To copy a file to another location and
   // overwrite the destination file if it already exists.
   System.IO.File.Copy(sourcepath, destpath, true);
   MessageBox.Show("copyFile success");



}
public enum ShowCommands : int
```

```
{
    SW_HIDE = 0,
    SW_SHOWNORMAL = 1,
    SW_NORMAL = 1,
    SW_SHOWMINIMIZED = 2,
    SW_SHOWMAXIMIZED = 3,
    SW_MAXIMIZE = 3,
    SW_SHOWNOACTIVATE = 4,
    SW_SHOW = 5,
    SW_MINIMIZE = 6,
    SW_SHOWMINNOACTIVE = 7,
    SW_SHOWNA = 8,
    SW_RESTORE = 9,
    SW_SHOWDEFAULT = 10,
    SW_FORCEMINIMIZE = 11,
    SW_MAX = 11
}
[DllImport("shell32.dll")]
static extern IntPtr ShellExecute(
    IntPtr hwnd,
    string lpOperation,
    string lpFile,
    string lpParameters,
    string lpDirectory,
```

```
        ShowCommands nShowCmd);



    private void button1_Click(object sender, EventArgs e)
    {
        //put the current docform to the a.txt
        if (this.doc == null )
            MessageBox.Show("Please choose one file or open a new file.");
        else
        {
            if (System.IO.File.Exists(System.Environment.CurrentDirectory +
"\\files\\ConsoleApplication1\\ConsoleApplication1\\ConsoleApplication1.obj"))
            {
              //  MessageBox.Show("delete consoleapp");
                System.IO.File.Delete(System.Environment.CurrentDirectory +
"\\files\\ConsoleApplication1\\ConsoleApplication1\\ConsoleApplication1.obj");
            }
            if (System.IO.File.Exists(System.Environment.CurrentDirectory +
"\\files\\ConsoleApplication1\\ConsoleApplication1\\result.txt"))
            {
              // MessageBox.Show("delete result");


                System.IO.File.Delete(System.Environment.CurrentDirectory +
"\\files\\ConsoleApplication1\\ConsoleApplication1\\result.txt");
            }
```

```
            if (System.IO.File.Exists(System.Environment.CurrentDirectory +
"\\files\\ConsoleApplication1\\ConsoleApplication1\\result.exe"))
            {
              //  MessageBox.Show("delete result.exe");
                System.IO.File.Delete(System.Environment.CurrentDirectory +
"\\files\\ConsoleApplication1\\ConsoleApplication1\\result.exe");
            }
            if (System.IO.File.Exists(System.Environment.CurrentDirectory +
"\\files\\PLT2\\result.cpp"))
            {
             //   MessageBox.Show("delete result.cpp!");
                System.IO.File.Delete(System.Environment.CurrentDirectory +
"\\files\\PLT2\\result.cpp");
            }
            try
            {
                SaveFileDialog sfd = new SaveFileDialog();
                DocForm df = (DocForm)this.doc;

                //  sfd.Filter = "文本文件（*.txt）|*.txt|RTF 文件|*.rtf|所有文件
（*.*）|*.*";

                sfd.FileName = System.Environment.CurrentDirectory +
"\\files\\PLT2\\a.txt";
                RichTextBoxStreamType fileType = TrunFileType(1);
                this.doc.SetFileTypeIndex(1);
```

```
            this.doc.SetFilePath(sfd.FileName);
            df.Sourse.SaveFile(sfd.FileName, fileType);

            df.SetFilePath(sfd.FileName);
            oldFileType = fileType;
           // MessageBox.Show("already saved ");
            //ouput the cpp file
            string path = System.Environment.CurrentDirectory +
"\\files\\PLT2\\codegen.exe";
            MessageBox.Show(path);

            if ((int)ShellExecute(Handle, "open", path, null,
System.Environment.CurrentDirectory + "\\files\\PLT2\\", 0) > 32)
MessageBox.Show("second success");
            if (!System.IO.File.Exists(System.Environment.CurrentDirectory +
"\\files\\PLT2\\result.cpp"))
                MessageBox.Show("code generation fails.There is no .cpp file
generated.");

            else
            {
                StreamReader sr = new
StreamReader(System.Environment.CurrentDirectory +
"\\files\\PLT2\\result.cpp");
                if (sr.ReadLine() == "ERROR")
```

```
                {

this.richTextBox1.GetType().GetProperty("Text").SetValue(this.richTextBox1,  "Fail! Error exists! Please check again!");
                }
                //move and substitute the cpp file
                copyFile(System.Environment.CurrentDirectory + "\\files\\PLT2\\result.cpp", System.Environment.
                CurrentDirectory + "\\files\\ConsoleApplication1\\ConsoleApplication1\\ConsoleApplication1.cpp");
                sr.Close();

                 System.Diagnostics.Process p = new System.Diagnostics.Process();
                 p.StartInfo.FileName = "cmd.exe";//要执行的程序名称
                 p.StartInfo.UseShellExecute = false;
                 p.StartInfo.RedirectStandardInput = true;//可能接受来自调用程序的输入信息
                 p.StartInfo.RedirectStandardOutput = true;//由调用程序获取输出信息
                 p.StartInfo.CreateNoWindow = true;//不显示程序窗口
                 p.StartInfo.Arguments = "/c cd files//ConsoleApplication1//ConsoleApplication1&&.\\VC\\vcvarsall.bat&&.\\Common7\\IDE\\devenv.com /RunExit ConsoleApplication1.vcxproj";

                 p.Start();//启动程序
```

```
            //produce the result.txt file
            /*  Process.Start("cmd.exe", "/c cd
files//ConsoleApplication1//ConsoleApplication1&&.\\VC\\vcvarsall.bat&&.\\VC\
\bin\\cl.exe"+
                  " stdafx.cpp ConsoleApplication1.cpp&&.\\VC\\bin\\link.exe /Fc
result.exe stdafx.obj ConsoleApplication1.obj");
               Process.Start("cmd.exe", "/c cd
files//ConsoleApplication1//ConsoleApplication1&&.\\VC\\vcvarsall.bat&&.\\Co
mmon7\\IDE\\devenv.com /runExit ConsoleApplication1.vcxproj");
            */
            //draw the picture
            string filepath = System.Environment.
            CurrentDirectory +
"\\files\\ConsoleApplication1\\ConsoleApplication1\\result.txt";
            while (!System.IO.File.Exists(filepath))
            {
               Console.WriteLine("in while"); MessageBox.Show("in while");
            }
            //    MessageBox.Show("out while");

            draw(filepath);
            //
(string)this.item3.GetType().GetProperty("Text").GetValue(this.item3);

this.richTextBox1.GetType().GetProperty("Text").SetValue(this.richTextBox1,
"success!");
         }
```

```
            }
            catch (Exception exce)
            {
                MessageBox.Show("there is an error! ");

this.richTextBox1.GetType().GetProperty("Text").SetValue(this.richTextBox1,
"fail!");
            }
        }

    }
    private void draw(string path)
    {


        if(!System.IO.File.Exists(path))
        {
            MessageBox.Show("The txt file does not exist! Compile again!");


        }


        else
        {
          // MessageBox.Show("before streamreader");
            StreamReader rd = new StreamReader(path);
```

```
            if(rd.ReadLine() == null)
            {
               MessageBox.Show("empty result.txt");
            }
            else{
            drawform = new DrawForm();
            drawform.Show();

            char[] delimiters = new char[] { ' ' };
            string line;
            while ((line = rd.ReadLine()) != null)
            {

               // MessageBox.Show(line);
               string[] parts = line.Split(delimiters,
StringSplitOptions.RemoveEmptyEntries);


               if (parts[0] == "Line")
               {
                  // MessageBox.Show("it is a line.x1 is " + parts[1] + " y1 is " + parts[2]
+ " x2 is " + parts[3] + " y2 is " + parts[4]);
                  drawform.drawLine(float.Parse(parts[1]), float.Parse(parts[2]),
float.Parse(parts[3]), float.Parse(parts[4]), 2);
                  //  drawform.drawLine(float.Parse(parts[1]), float.Parse(parts[2]),
float.Parse(parts[3]), float.Parse(parts[4]),int.Parse(parts[5]));
```

```
            }
            else if (parts[0] == "Circle")
            {
               drawform.drawCircle(float.Parse(parts[1]), float.Parse(parts[2]),
float.Parse(parts[3]), 1);
            }
            else if (parts[0] == "Point")
            {
              // MessageBox.Show("point: x: " + parts[1] + " y: " + parts[2]);
               drawform.drawPoint(float.Parse(parts[1]), float.Parse(parts[2]));

            }
            else if (parts[0] == "END")
            {
               break;
            }
            else { }

         }
         }
         rd.Close();
      }
   }
```

```
        private void tsmi_Edit_Click(object sender, EventArgs e)
        {

        }

        private void tsmi_Help_Click(object sender, EventArgs e)
        {

        }

        private void MainForm_Load(object sender, EventArgs e)
        {

        }

        private void item2_Click(object sender, EventArgs e)
        {
            string path =
(string)this.item3.GetType().GetProperty("Text").GetValue(this.item2);

            showText(path);
        }

        private void item3_Click(object sender, EventArgs e)
        {
```

```
        string path =
(string)this.item3.GetType().GetProperty("Text").GetValue(this.item3);

        showText(path);

    }

    private void showText(string path)
    {
        StreamReader rd = new StreamReader(path);
        String str = rd.ReadToEnd();

        openFileDialog1.FileName = path;
        RichTextBoxStreamType fileType =
TrunFileType(openFileDialog1.FilterIndex);
        //  MessageBox.Show("file name is " + openFileDialog1.FileName);
        ///  wCount++;
        doc = new DocForm(RichTextBoxStreamType.PlainText,
openFileDialog1.FileName, openFileDialog1.FilterIndex);
        doc.MdiParent = this;
        doc.Show();
```

```
            listDocForm.Add(doc);
            doc.WindowState = FormWindowState.Maximized;

        }









    }
}
```

6. Program.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using System.Windows.Forms;
//Download by http://www.codefans.net
```

```
namespace 多文档编辑器
{
    static class Program
    {
        /// <summary>
        /// 应用程序的主入口点。
        /// </summary>
        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            MainForm form1 = new MainForm();
           // DocForm form2 = new DocForm() ;
           // DrawForm form2 = new DrawForm();
         //  MainForm form2 = new MainForm();
            form1.Show(); //FormWindowState.
            form1.WindowState = FormWindowState.Maximized;


            Application.Run();


        }
    }
}
```