编译原理课程设计

目录

概况

小组成员(不分先后)： xxx yyy

成员分工： 5：5

实现语言： C# (IDE)
C++ (Compiler)

总共耗时： 一个月

awesomeCC awesome C# Compiler

awesomeCC awesome C# Compiler

Compiler 实现：C++ （目前5k行的样子）

使用CMake编译（前后端分离，编译出来的解释器小）

使用Google Test 进行单元测试

使用travis.CI 进行集成测试

愉快的合作： doxygen 注释规范 + Google Code Style Guide

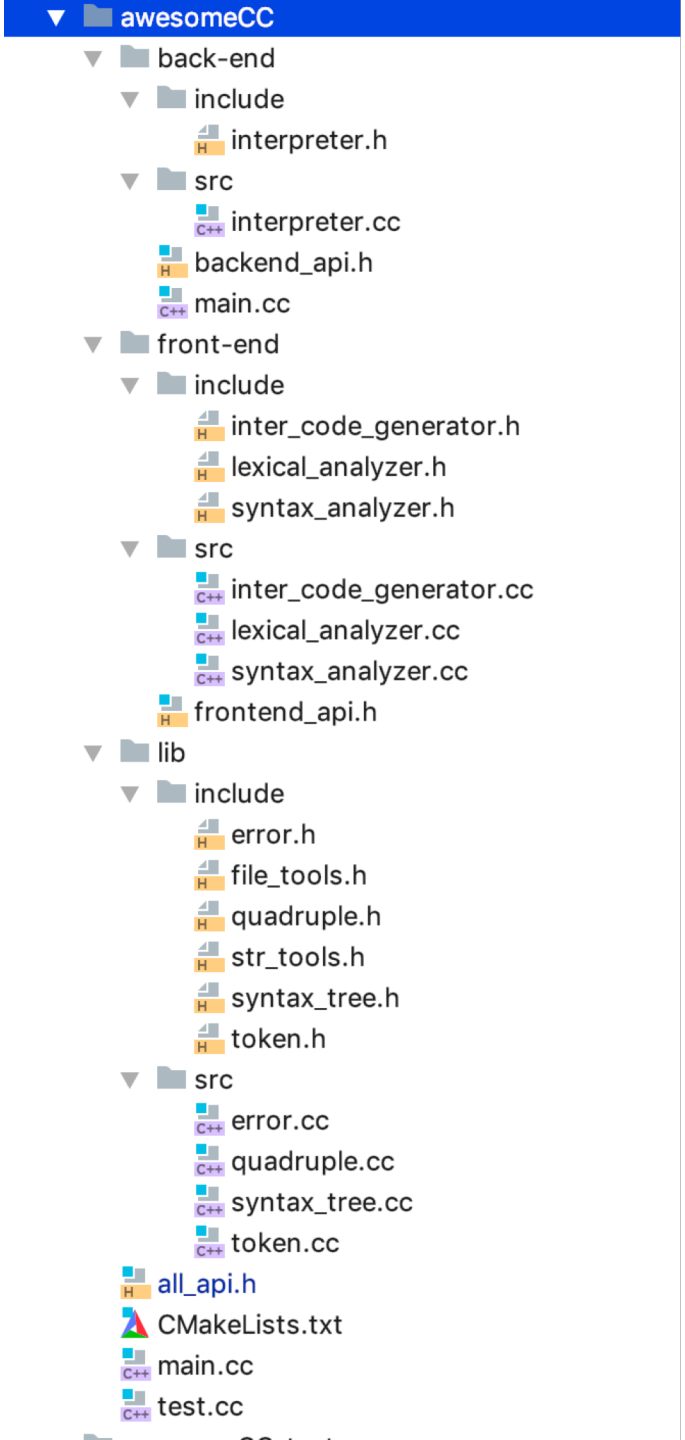
Language	files	blank	comment	code
C++	13	521	576	2426
C/C++ Header	13	174	193	542
CMake	20	90	63	449
SUM:	46	785	832	3417

项目结构

后端
解释器

前端
词法分析
语法分析
语义分析&中间代码生成

自己的库
错误
文件读取
四元式
常用的字符串函数
Token



awesomeCC 所有功能

像gcc学习😝（这是命令行使用acc的方式）

```
~/Workspace/CompilePrinciple/awesomeCC/cmake-build-debug/awesomeCC  master  ./acc --version
awesome CC compiler 1.0
developed by Keyi Li & Hanwen Liu
~/Workspace/CompilePrinciple/awesomeCC/cmake-build-debug/awesomeCC  master  ./acc --help
OVERVIEW: awesome CC compiler, developed by Keyi Li & Hanwen Liu

USAGE: acc file_name [options] <inputs>

OPTIONS:
-a, --assembler              generate inter code(Quadruple) for a source AC file
-h, --help                   get help on awesomeCC command line arguments
-i, --interpreter            interpret and execute an inter code file
-l, --lexer                  lexical analyze a source AC file
-o, --output                 the output file path
-p, --parser                 syntax analyze a source AC file
-v, --version                display awesomeCC version

EXAMPLE:
acc source.ac -l
acc source.ac -p
acc source.ac -a
acc source.ac.ic -i
acc source.ac
acc -h
acc -v
```

awesomeCC 词法分析

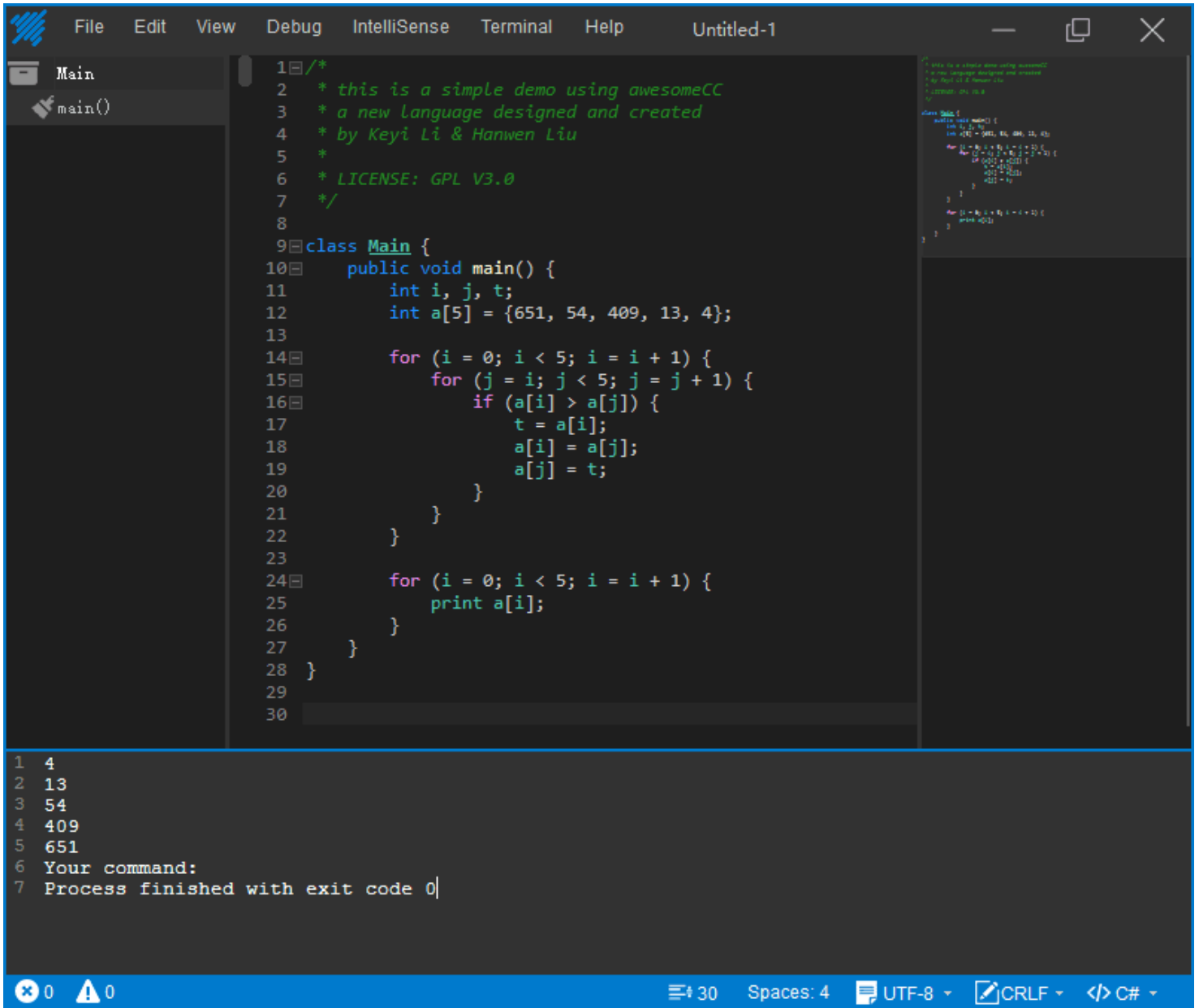
模拟DFA

生成token列表

我们能够直接匹配到的关键字、操作符、分隔符

```
77  vector<string> Token::KEYWORDS = {
78          "int", "float", "double", "char", "void", "class", "public",
79          "if", "for", "while", "do", "else",
80          "include", "return", "print"
81  };
82
83
84  vector<string> Token::OPERATORS = {"+",  "-",  "<",  ">", "!", "=", "||", "&&", "*", "/", "%",
85                                     "++", "--", "<<", ">>",
86                                     "<=", ">=", "!=", "=="};
87
88
89  vector<char> Token::SEPARATORS = {'(', ')', '{', '}', '[', ']', ',', '\'', ';' , '\'', '\"'};
90
```

awesomeCC 词法分析 一个例子☝



awesomeCC 词法分析 一个例子☝

```
~/Workspace/CompilePrinciple/awesomeCC/demo  master ●  ./acc bubble_sort.ac -l
Tokens
     class    type: keyword
      Main    type: identifier
         {    type: separator
    public    type: keyword
      void    type: keyword
      main    type: identifier
         (    type: separator
         )    type: separator
         {    type: separator
       int    type: keyword
         i    type: identifier
         ,    type: separator
         j    type: identifier
         ,    type: separator
         t    type: identifier
         ;    type: separator
       int    type: keyword
         a    type: identifier
         [    type: separator
         5    type: digit constant
         ]    type: separator
         =    type: operator
         {    type: separator
       651    type: digit constant
         ,    type: separator
        54    type: digit constant
         ,    type: separator
       409    type: digit constant
         ,    type: separator
        13    type: digit constant
         ,    type: separator
         4    type: digit constant
         }    type: separator
         ;    type: separator
       for    type: keyword
         (    type: separator
         i    type: identifier
         =    type: operator
         0    type: digit constant
         ;    type: separator
         i    type: identifier
         <    type: operator
         5    type: digit constant
         ;    type: separator
         i    type: identifier
```

awesomeCC 语法分析

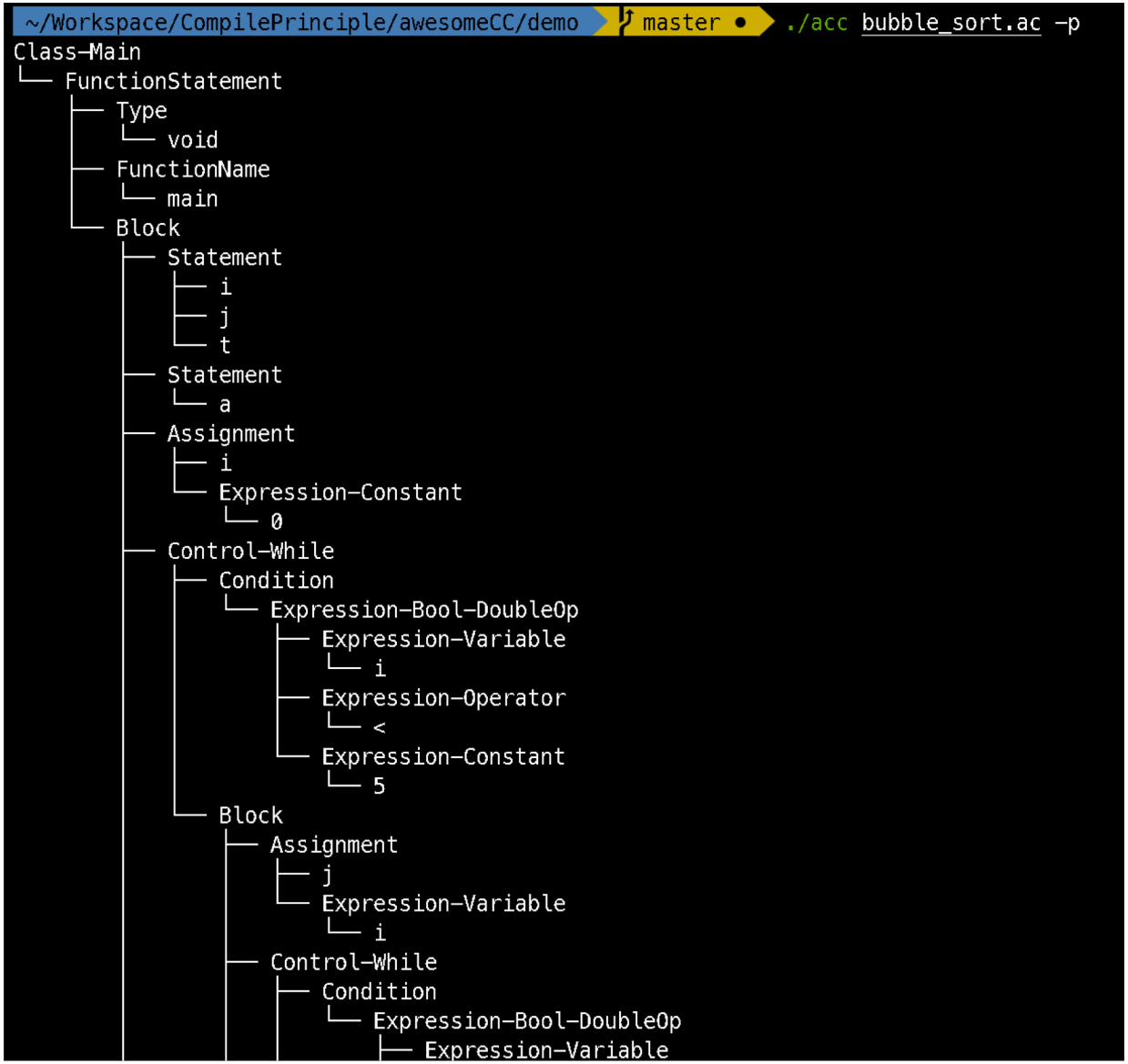
递归下降 LL(k)

生成语法树🌲

我们能识别的语句类型

```
void _print(SyntaxTreeNode * father_node);
void _include(SyntaxTreeNode * father_node);
void _functionStatement(SyntaxTreeNode * father_node);
void _statement(SyntaxTreeNode * father_node);
void _functionCall(SyntaxTreeNode * father_node);
void _block(SyntaxTreeNode * father_node);
void _return(SyntaxTreeNode * father_node);
void _expression(SyntaxTreeNode * father_node, TOKEN_TYPE_ENUM stop_token = TOKEN_TYPE_ENUM::SEMICOLON);
void _assignment(SyntaxTreeNode * father_node, TOKEN_TYPE_ENUM stop_token = TOKEN_TYPE_ENUM::SEMICOLON);
void _control(SyntaxTreeNode * father_node);
void _for(SyntaxTreeNode * father_node);
void _while(SyntaxTreeNode * father_node);
void _if(SyntaxTreeNode * father_node);
void _else(SyntaxTreeNode * father_node);
void _else_if(SyntaxTreeNode * father_node);
```

awesomeCC 语法分析 还是那个例子☝



awesomeCC 语义分析 & 中间代码生成

语法制导翻译
dfs语法树的时候进行翻译

四元式！（解释执行起来简单，和runtime一起打包生成可执行文件小）

```
enum class INTER_CODE_OP_ENUM {
    /* arithmetic */
    ADD,
    SUB,
    DIV,
    MUL,
    MOD,
    /* jump */
    J,    // 啥都不管 直接跳
    JE,   // arg1 == arg2 跳转
    JNE,  // arg1 == arg2 跳转
    JL,   // arg1 < arg2 跳转
    JG,   // arg1 > arg2 跳转
    /* other */
    MOV,  // 赋值
    PRINT // 输出
};
```

awesomeCC 语义分析 & 中间代码生成

Condition树，拉链回填（backpatch）

维护next_list, true_list, false_list

1) $S \rightarrow \textbf{if}\ (\ B\ )\ M\ S_1$ { *backpatch*($B.truelist$, $M.instr$);
   $S.nextlist = merge(B.falselist, S_1.nextlist)$; }

2) $S \rightarrow \textbf{if}\ (\ B\ )\ M_1\ S_1\ N\ \textbf{else}\ M_2\ S_2$
   { *backpatch*($B.truelist$, $M_1.instr$);
   backpatch($B.falselist$, $M_2.instr$);
   $temp = merge(S_1.nextlist, N.nextlist)$;
   $S.nextlist = merge(temp, S_2.nextlist)$; }

3) $S \rightarrow \textbf{while}\ M_1\ (\ B\ )\ M_2\ S_1$
   { *backpatch*($S_1.nextlist$, $M_1.instr$);
   backpatch($B.truelist$, $M_2.instr$);
   $S.nextlist = B.falselist$;
   gen('goto' $M_1.instr$); }

4) $S \rightarrow \{\ L\ \}$ { $S.nextlist = L.nextlist$; }

5) $S \rightarrow A\ ;$ { $S.nextlist = \textbf{null}$; }

6) $M \rightarrow \epsilon$ { $M.instr = nextinstr$; }

7) $N \rightarrow \epsilon$ { $N.nextlist = makelist(nextinstr)$;
   gen('goto _'); }

8) $L \rightarrow L_1\ M\ S$ { *backpatch*($L_1.nextlist$, $M.instr$);
   $L.nextlist = S.nextlist$; }

9) $L \rightarrow S$ { $L.nextlist = S.nextlist$; }

1) $B \rightarrow B_1\ ||\ M\ B_2$ { *backpatch*($B_1.falselist$, $M.instr$);
   $B.truelist = merge(B_1.truelist, B_2.truelist)$;
   $B.falselist = B_2.falselist$; }

2) $B \rightarrow B_1\ \&\&\ M\ B_2$ { *backpatch*($B_1.truelist$, $M.instr$);
   $B.truelist = B_2.truelist$;
   $B.falselist = merge(B_1.falselist, B_2.falselist)$; }

3) $B \rightarrow !\ B_1$ { $B.truelist = B_1.falselist$;
   $B.falselist = B_1.truelist$; }

4) $B \rightarrow (\ B_1\ )$ { $B.truelist = B_1.truelist$;
   $B.falselist = B_1.falselist$; }

5) $B \rightarrow E_1\ \textbf{rel}\ E_2$ { $B.truelist = makelist(nextinstr)$;
   $B.falselist = makelist(nextinstr + 1)$;
   gen('if' $E_1.addr$ **rel**.op $E_2.addr$ 'goto _');
   gen('goto _'); }

6) $B \rightarrow \textbf{true}$ { $B.truelist = makelist(nextinstr)$;
   gen('goto _'); }

7) $B \rightarrow \textbf{false}$ { $B.falselist = makelist(nextinstr)$;
   gen('goto _'); }

8) $M \rightarrow \epsilon$ { $M.instr = nextinstr$; }

awesomeCC 语义分析 & 中间代码生成

```
~/Workspace/CompilePrinciple/awesomeCC/demo  master  ./acc bubble_sort.ac -a
Generated 29 inter codes
#0  (MOV   , 651   ,        , v3[0] )
#1  (MOV   , 54    ,        , v3[1] )
#2  (MOV   , 409   ,        , v3[2] )
#3  (MOV   , 13    ,        , v3[3] )
#4  (MOV   , 4     ,        , v3[4] )
#5  (MOV   , 0     ,        , v0    )
#6  (JL    , v0    , 5      , 8     )
#7  (J     ,       ,        , 22    )
#8  (MOV   , v0    ,        , v1    )
#9  (JL    , v1    , 5      , 11    )
#10 (J     ,       ,        , 19    )
#11 (JG    , v3[v0], v3[v1], 13    )
#12 (J     ,       ,        , 16    )
#13 (MOV   , v3[v0],        , v2    )
#14 (MOV   , v3[v1],        , v3[v0])
#15 (MOV   , v2    ,        , v3[v1])
#16 (ADD   , v1    , 1      , t0    )
#17 (MOV   , t0    ,        , v1    )
#18 (J     ,       ,        , 9     )
#19 (ADD   , v0    , 1      , t1    )
#20 (MOV   , t1    ,        , v0    )
#21 (J     ,       ,        , 6     )
#22 (MOV   , 0     ,        , v0    )
#23 (JL    , v0    , 5      , 25    )
#24 (J     ,       ,        , 29    )
#25 (PRINT , v3[v0],        ,       )
#26 (ADD   , v0    , 1      , t2    )
#27 (MOV   , t2    ,        , v0    )
#28 (J                      , 23    )
```

awesomeCC 中间代码的解释执行

堆区、栈区

```
class Interpreter {
private:
    int index; // 我的pc指针
    vector<Quadruple> code;
    vector<double> t_stack;
    vector<double> v_stack;
```

awesomeCC 中间代码的解释执行

立即数寻址、相对寻址、相对变址寻址

```
double Interpreter::_getValue(string value_str) {
    // 寻址
    if (value_str[0] == 'v') {
        int len = value_str.size();
        for (int i = 0; i < len; i ++)
            if (value_str[i] == '[') {
                // 相对寻址 || 相对变址寻址
                int offset = _getValue(value_str.substr(i + 1, len - i - 2));
                int base = _getValue(value_str.substr(1, i - 1));
                return v_stack[offset + base];
            }

        // 立即数寻址
        int temp_index = string2int(value_str.substr(1));
        return v_stack[temp_index];
    }
    // 临时变量
    if (value_str[0] == 't') {
        int temp_index = string2int(value_str.substr(1));
        return t_stack[temp_index];
    }
    // 立即数
    else
        return string2double(value_str);
}
```

awesomeCC 中间代码的解释执行

```
~/Workspace/CompilePrinciple/awesomeCC/demo  master  ./acc bubble_sort.ac.ic -i
4
13
54
409
651
```

awesomeCC 总的来一遍？当然可以了！

编译 && 执行

```
~/Workspace/CompilePrinciple/awesomeCC/demo  master ●  ./acc bubble_sort.ac
start compiling bubble_sort.ac...

compile finish in 0 sec(s).

------ start executing ------
4
13
54
409
651
```

awesomeCC

命令行太geek了 来点好看的吧？

IDE

100% in C#

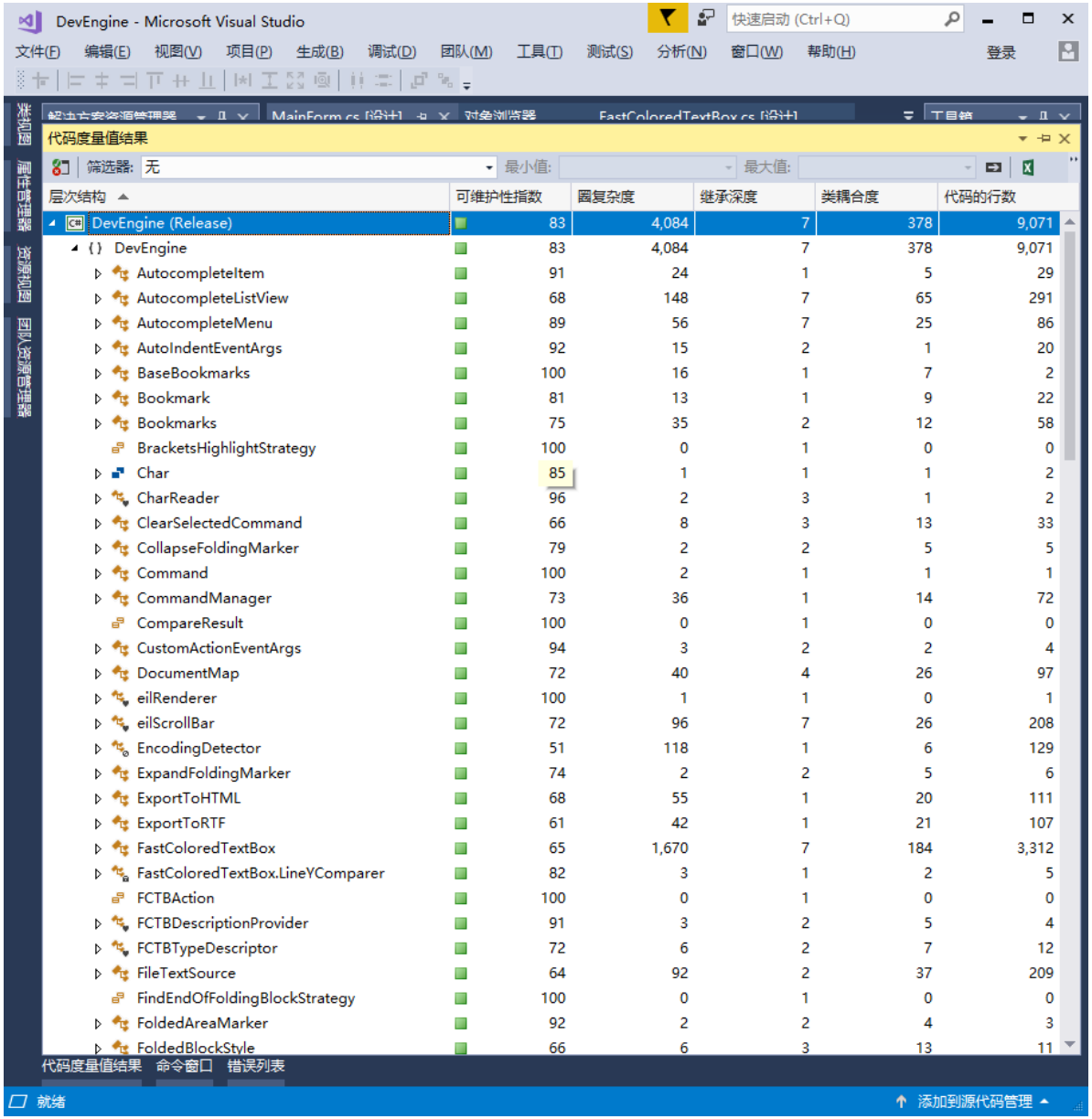
运行环境 Windows 7 or Higher

所有用到的第三方库：

FCTB https://github.com/PavelTorgashov/FastColoredTextBox

IDE

代码统计：9k+（不含GUI代码，含FCTB代码，约占一半）



层次结构	可维护性指数	圈复杂度	继承深度	类耦合度	代码的行数
DevEngine (Release)	83	4,084	7	378	9,071
DevEngine	83	4,084	7	378	9,071
AutocompleteItem	91	24	1	5	29
AutocompleteListView	68	148	7	65	291
AutocompleteMenu	89	56	7	25	86
AutoIndentEventArgs	92	15	2	1	20
BaseBookmarks	100	16	1	7	2
Bookmark	81	13	1	9	22
Bookmarks	75	35	2	12	58
BracketsHighlightStrategy	100	0	1	0	0
Char	85	1	1	1	2
CharReader	96	2	3	1	2
ClearSelectedCommand	66	8	3	13	33
CollapseFoldingMarker	79	2	2	5	5
Command	100	2	1	1	1
CommandManager	73	36	1	14	72
CompareResult	100	0	1	0	0
CustomActionEventArgs	94	3	2	2	4
DocumentMap	72	40	4	26	97
eilRenderer	100	1	1	0	1
eilScrollBar	72	96	7	26	208
EncodingDetector	51	118	1	6	129
ExpandFoldingMarker	74	2	2	5	6
ExportToHTML	68	55	1	20	111
ExportToRTF	61	42	1	21	107
FastColoredTextBox	65	1,670	7	184	3,312
FastColoredTextBox.LineYComparer	82	3	1	2	5
FCTBAction	100	0	1	0	0
FCTBDescriptionProvider	91	3	2	5	4
FCTBTypeDescriptor	72	6	2	7	12
FileTextSource	64	92	2	37	209
FindEndOfFoldingBlockStrategy	100	0	1	0	0
FoldedAreaMarker	92	2	2	4	3
FoldedBlockStyle	66	6	3	13	11

IDE – C#

C#是微软推出的一种基于.NET框架的、面向对象的高级编程语言。

C#以.NET框架类库作为基础，拥有类似Visual Basic的快速开发能力。

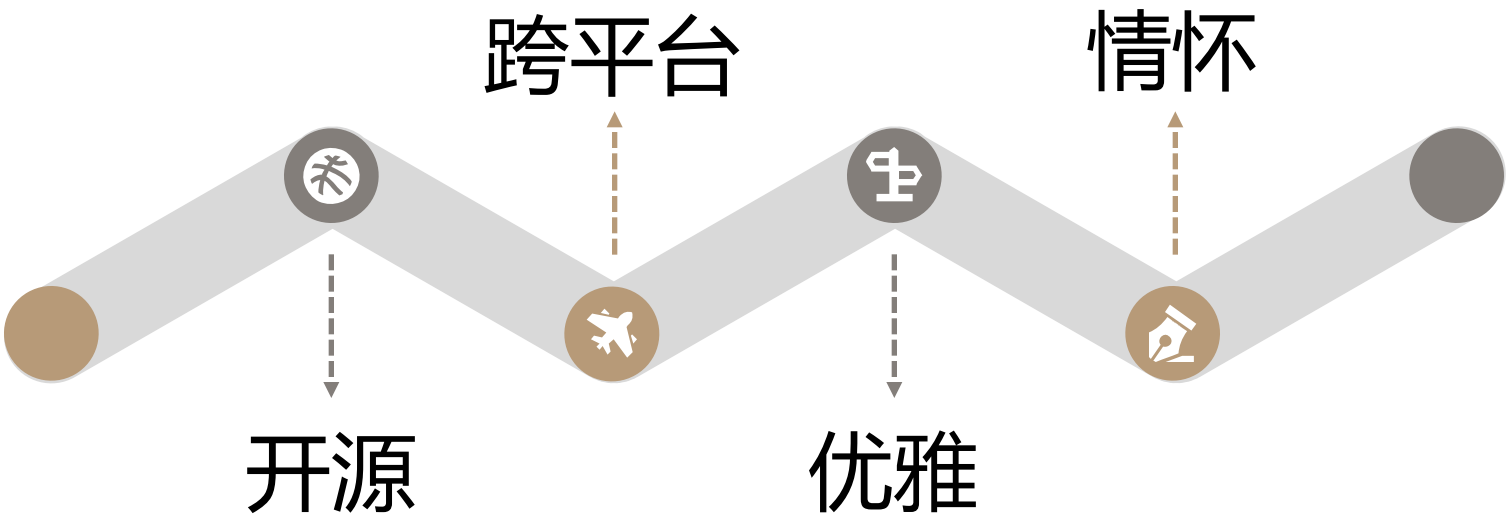
C#由安德斯·海尔斯伯格主持开发，微软在2000年发布了这种语言，希望借助这种语言来取代Java。

C#已经成为Ecma国际和国际标准组织的标准规范。

暂不讨论其他特性，

如果说C++是“带类的C”的话，我认为C#就是“只带类的C”。

IDE – Why C#



IDE – Why Not C#

QT？编译出来的程序太大了吧！

Electron？（Huge Runtime）太慢了吧！况且已经有VS Code了！

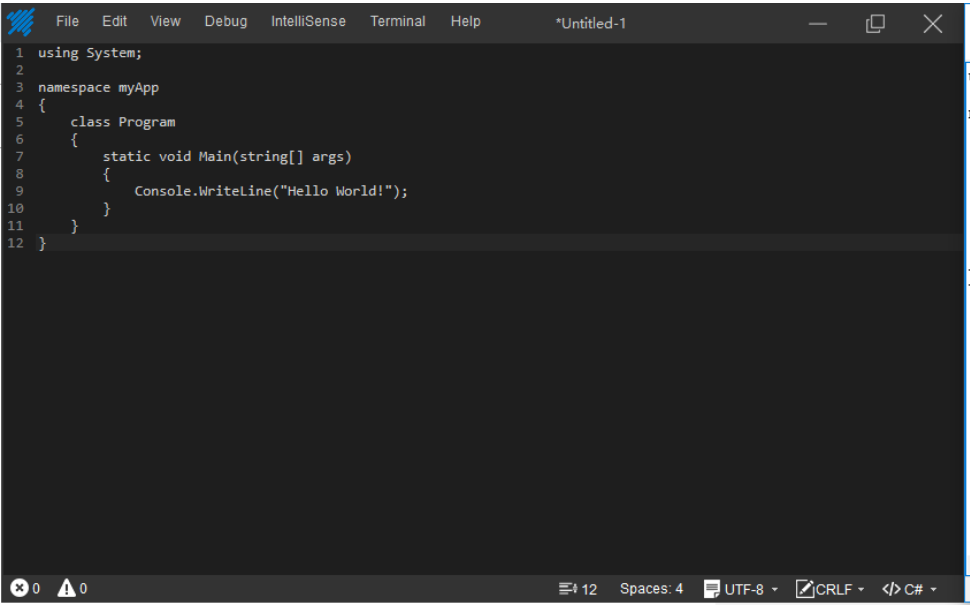
易语言？太容易被杀毒软件误报了吧！

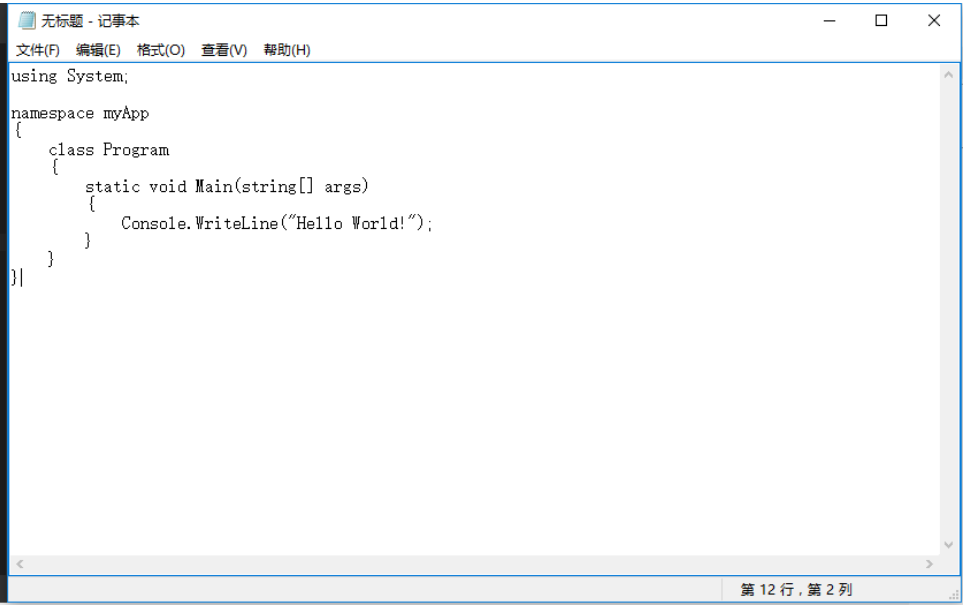
C++ with自研GUI？成本太高了吧！

也许C#不是世界上最好的语言，但它对于我们的课设来说是最好的。

IDE – 记事本？

《大家来找茬》，找找看，下面两个程序有哪些不同？

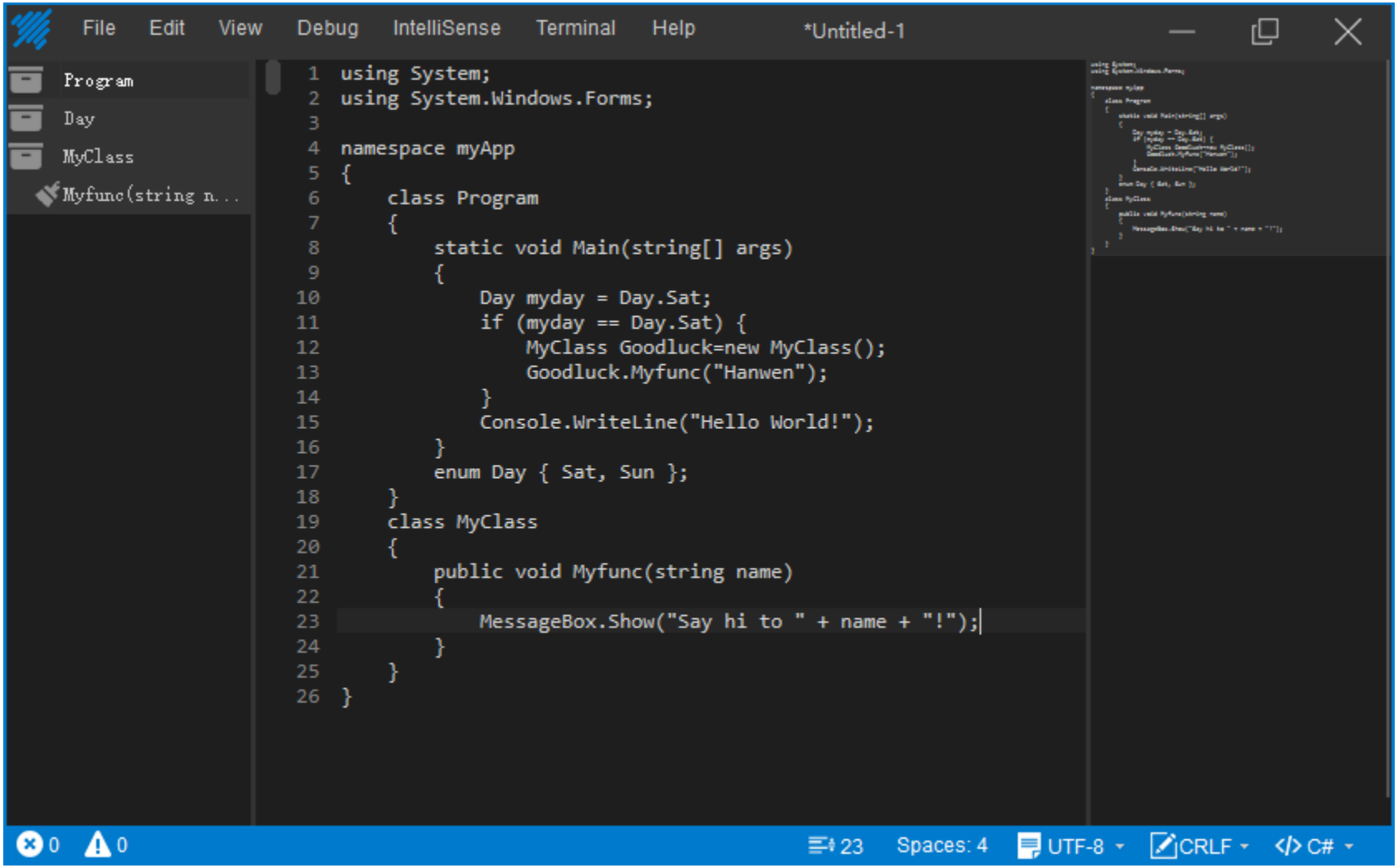




不就是一个换了个配色的Notepad 2.0嘛？

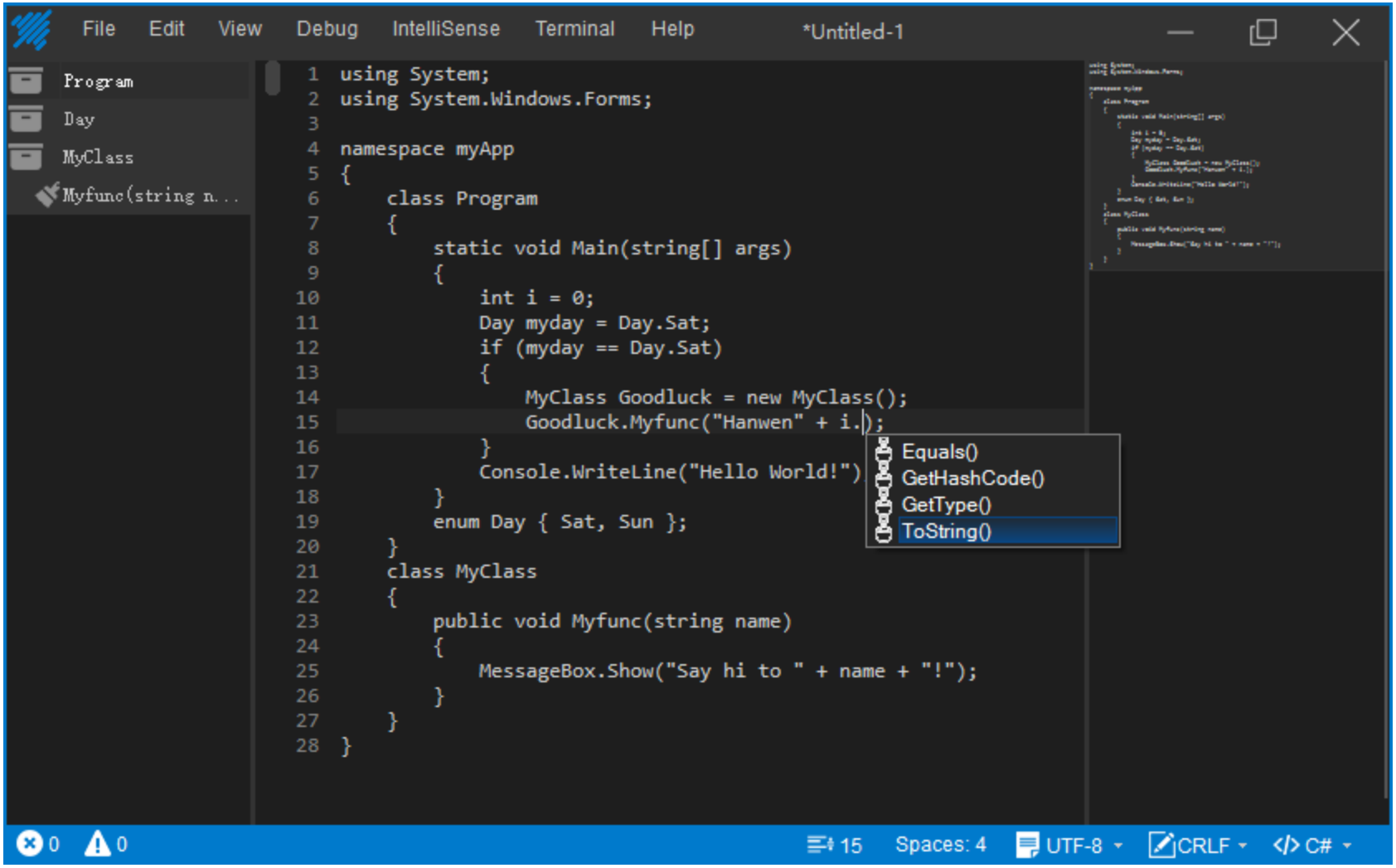
IntelliSense – Structure Sense

代码结构？IDE更懂你



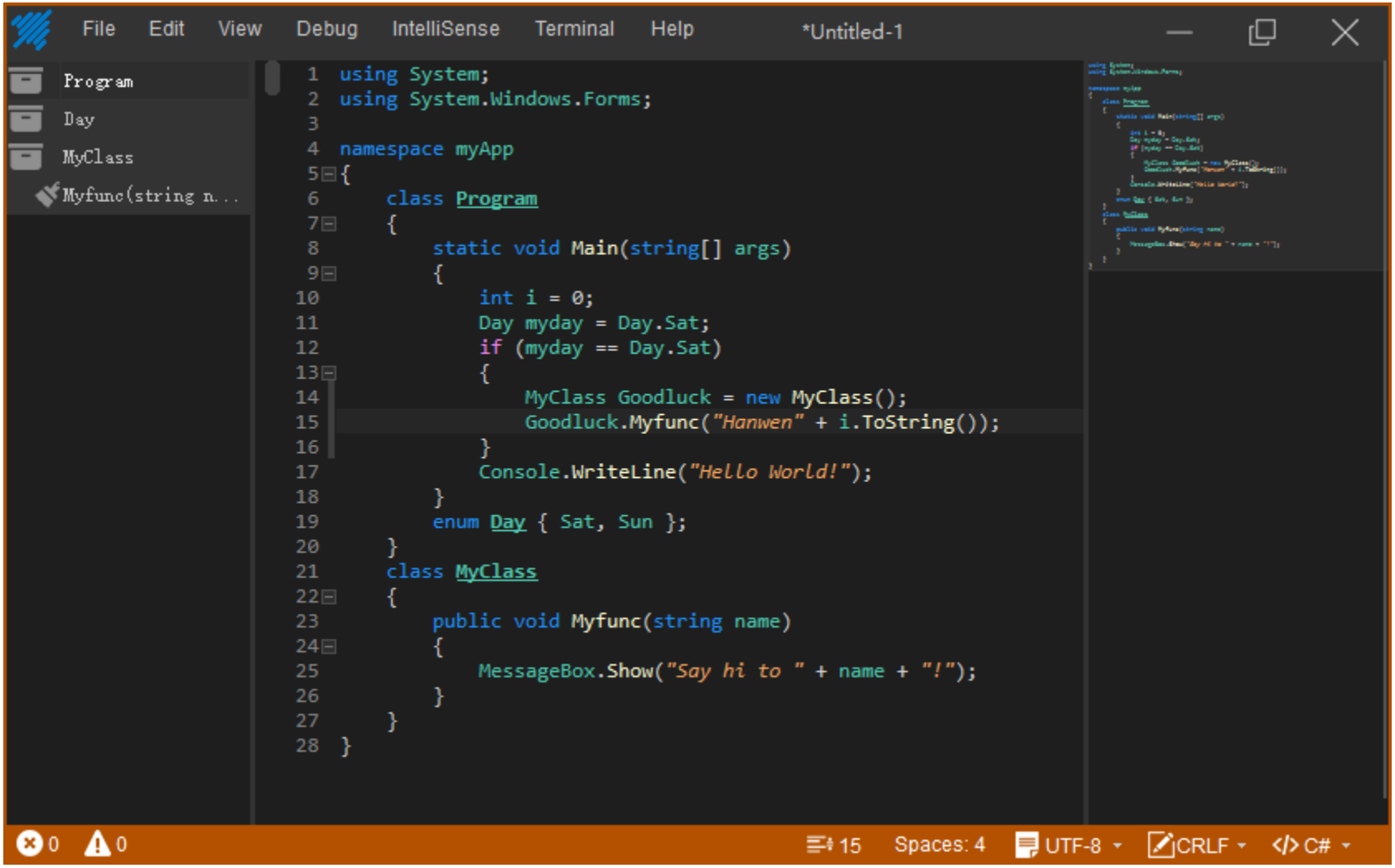
IntelliSense – Coding Sense

你的第二大脑



IntelliSense – Color Sense

世界不能失去色彩



IntelliSense – File Sense

数据无价，不再惧怕突然断电