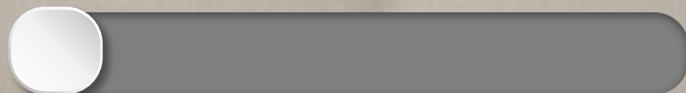


编译原理课程设计



目录

概述

01

编译器

03

总结

05

02

Awesome C#

04

IDE

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

Nov 4, 2018 – Dec 9, 2018

Contributions: Commits ▾

Contributions to master, excluding merge commits



awesomeCC

awesome C# Compiler



awesomeCC

awesome C# Compiler

Compiler 实现: C++ (目前5k行的样子)

使用CMake编译 (前端后端分离, 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

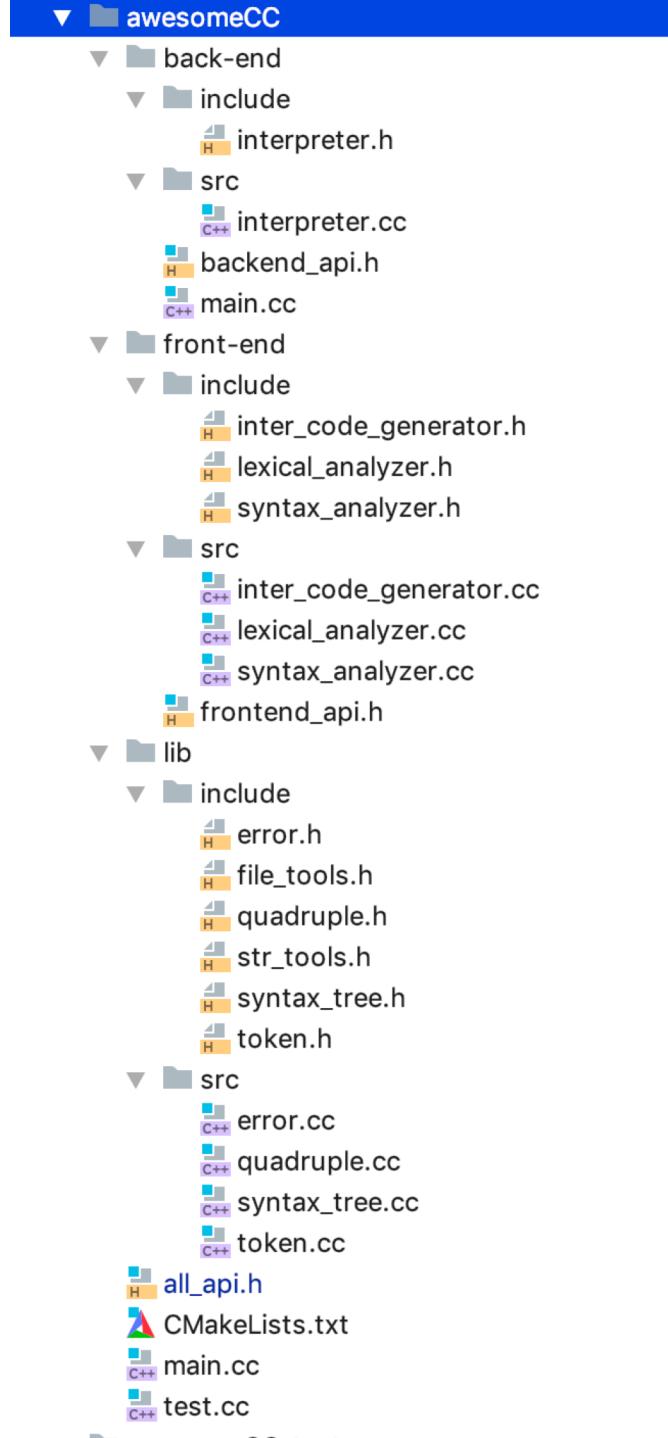
Language	files	blank	comment	code
C++	13	521	576	2426
C/C++ Header	13	174	193	542
CMake	20	90	63	449
SUM:	46	785	832	3417

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:

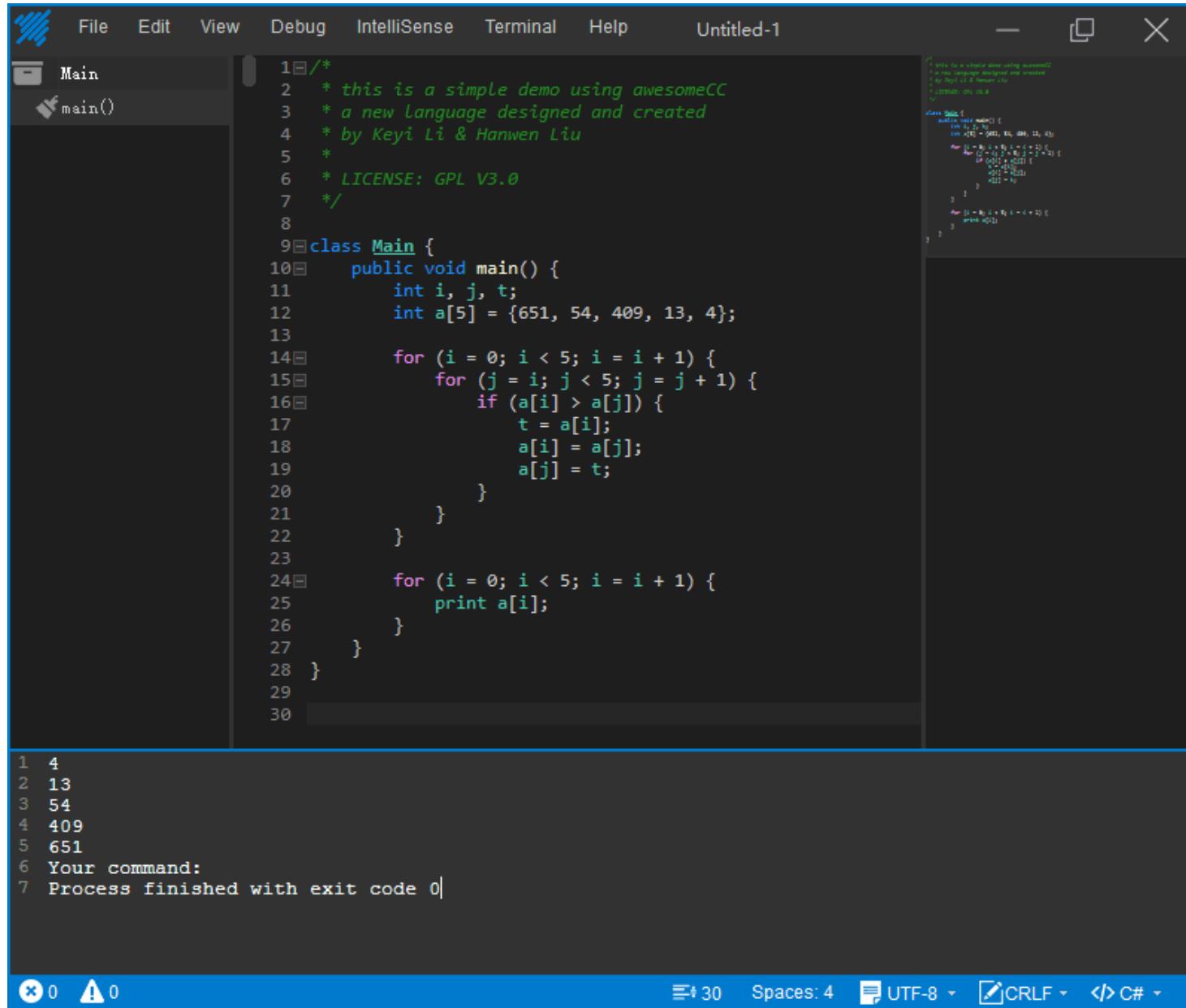
```
acc source.ac -l  
acc source.ac -p  
acc source.ac -a  
acc source.ac.ic -i  
acc source.ac  
acc -h  
acc -v
```

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      ',', '.', '\n', ';', '\t', '\v'}
```



The screenshot shows a code editor interface with a dark theme. On the left, a sidebar displays a file tree with a single entry: 'Main'. Below it, a code editor window shows a C-like program:

```
1 /*  
2  * this is a simple demo using awesomeCC  
3  * a new language designed and created  
4  * by Keyi Li & Hanwen Liu  
5  *  
6  * LICENSE: GPL V3.0  
7 */  
8  
9 class Main {  
10    public void main() {  
11        int i, j, t;  
12        int a[5] = {651, 54, 409, 13, 4};  
13  
14        for (i = 0; i < 5; i = i + 1) {  
15            for (j = i; j < 5; j = j + 1) {  
16                if (a[i] > a[j]) {  
17                    t = a[i];  
18                    a[i] = a[j];  
19                    a[j] = t;  
20                }  
21            }  
22        }  
23  
24        for (i = 0; i < 5; i = i + 1) {  
25            print a[i];  
26        }  
27    }  
28 }
```

The right side of the interface shows the output of the program's execution:

```
1 4  
2 13  
3 54  
4 409  
5 651  
6 Your command:  
7 Process finished with exit code 0
```

At the bottom, there are several status indicators: a red 'x' icon, a yellow triangle icon, '30' (likely referring to tabs or windows), 'Spaces: 4', 'UTF-8', 'CRLF', and '</> C#'. There is also a small logo in the bottom left corner.

```
~/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
  ,
  j       type: identifier
  ,
  t       type: identifier
  ;
  int     type: keyword
  a       type: identifier
  [
  5       type: digit constant
  ]
  =
  {
  651    type: digit constant
  ,
  54     type: digit constant
  ,
  409   type: digit constant
  ,
  13    type: digit constant
  ,
  4     type: digit constant
  }
  ;
  for   type: keyword
  (
  i     type: identifier
  =
  0     type: digit constant
  ;
  i     type: identifier
  <
  5     type: digit constant
  ;
  i     type: identifier
```

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 🤝

```
~/Workspace/CompilePrinciple/awesomeCC/demo(master) ➜ ./acc bubble_sort.ac -p
Class-Main
└ FunctionStatement
  └ Type
    └ void
  └ FunctionName
    └ main
  └ Block
    └ Statement
      └ i
      └ j
      └ t
    └ Statement
      └ a
    └ Assignment
      └ i
      └ Expression-Constant
        └ 0
    └ Control-While
      └ Condition
        └ Expression-Bool-DoubleOp
          └ Expression-Variable
            └ i
          └ Expression-Operator
            └ <
          └ Expression-Constant
            └ 5
    └ Block
      └ Assignment
        └ j
        └ Expression-Variable
          └ i
      └ Control-While
        └ Condition
          └ Expression-Bool-DoubleOp
            └ Expression-Variable
```

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
};
```

Condition树，拉链回填 (backpatch)

维护next_list, true_list, false_list

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

```
~/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 )
```

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;
```

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);
}
```

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

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
```

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)

The screenshot shows the Microsoft Visual Studio interface with the title bar "DevEngine - Microsoft Visual Studio". The menu bar includes File, Edit, View, Project, Build, Debug, Team, Tools, Test, Analyze, Window, and Help. A login button is also present in the top right. The main window displays the "Code Metrics Results" window, which lists various code components with their maintainability index, coupling, and complexity metrics. The results are grouped by project, with the first item being "DevEngine (Release)".

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.LineVComparer	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。

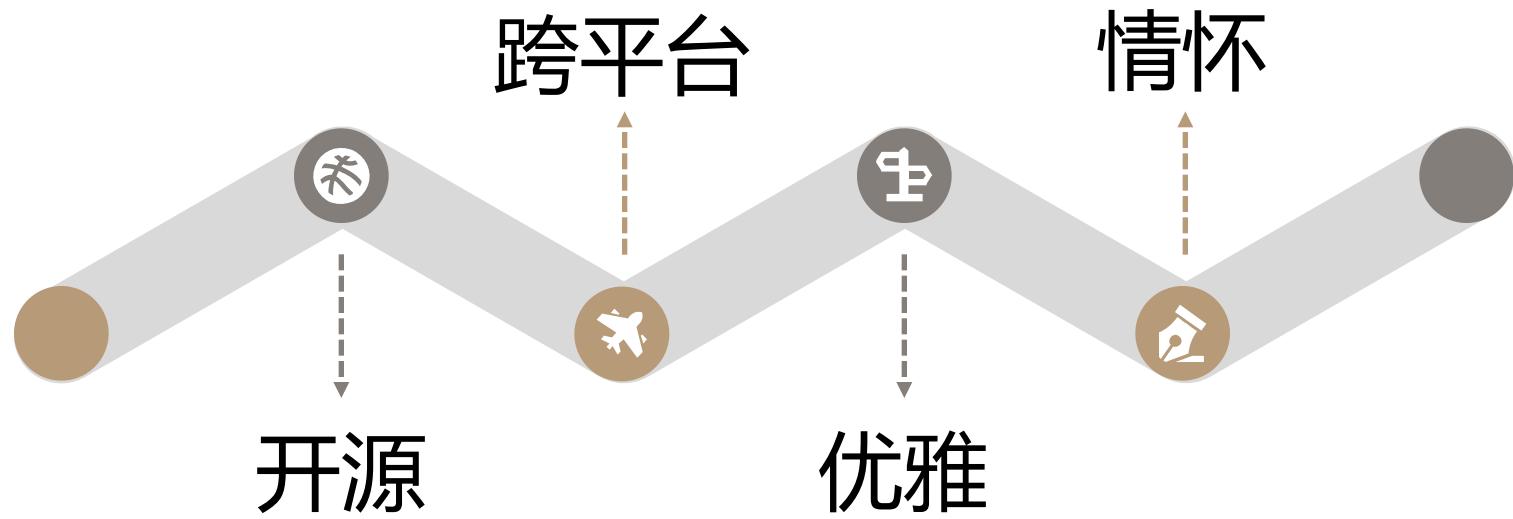
C#由安德斯·海尔斯伯格主持开发，微软在2000年发布了这种语言，希望借助这种语言来取代Java。

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 !

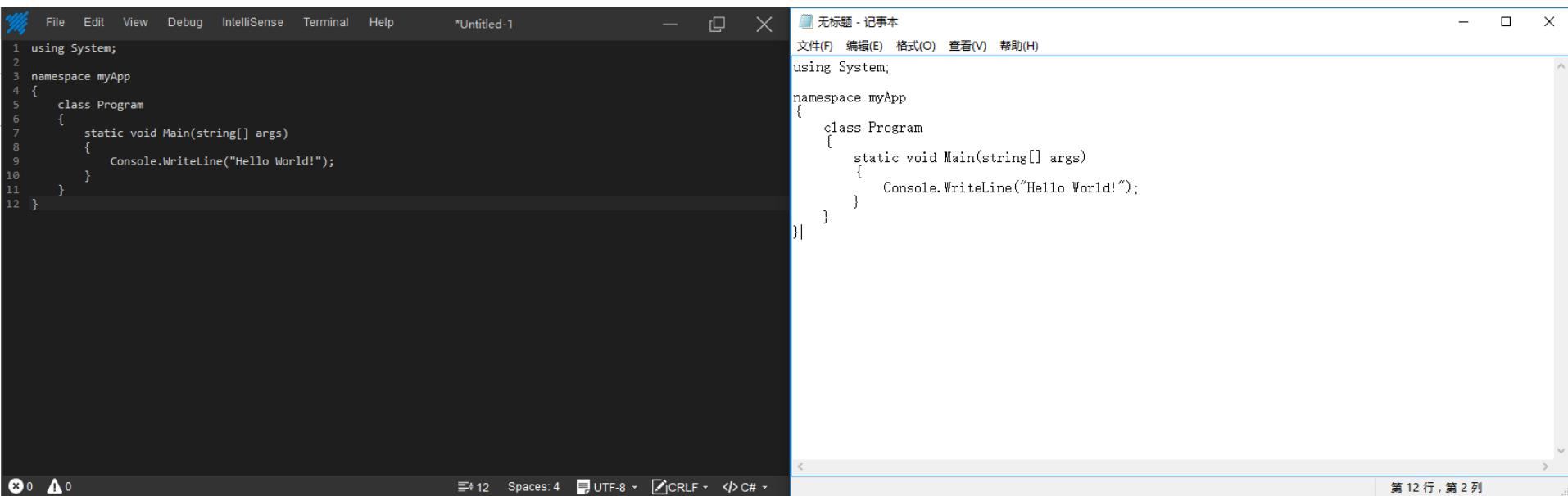
易语言 ? 太容易被杀毒软件误报了吧 !

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？



The screenshot shows a Windows Notepad window with two tabs open. The left tab, titled "Untitled-1", contains the following C# code:

```
1 using System;
2
3 namespace myApp
4 {
5     class Program
6     {
7         static void Main(string[] args)
8         {
9             Console.WriteLine("Hello World!");
10        }
11    }
12 }
```

The right tab, titled "无标题 - 记事本", contains the same C# code:

```
无标题 - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
using System;

namespace myApp
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Hello World!");
        }
    }
}
```

The Notepad interface includes a menu bar with File, Edit, View, Debug, IntelliSense, Terminal, Help, and Untitled-1. At the bottom, there are status bars for file count (0), line count (12), spaces (4), encoding (UTF-8), line endings (CRLF), and language (C#). A cursor is visible at the end of the second line of the right-hand code.

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

The screenshot shows a dark-themed IDE interface with the following details:

- Menu Bar:** File, Edit, View, Debug, IntelliSense, Terminal, Help.
- Code Editor:** The main window displays C# code. The code defines a `Program` class with a `Main` method and an `MyClass` class with a `Myfunc` method. It also defines an `Day` enum with values `Sat` and `Sun`.
- Toolbars and Status Bar:** The bottom status bar shows icons for errors (0), warnings (0), spaces (Spaces: 4), encoding (UTF-8), line endings (CRLF), and file type (C#).
- Code Completion:** A tooltip or completion list is visible on the right side of the editor, showing the current line of code and the available completions for the variable `name`.
- Syntax Highlighting:** The code is syntax-highlighted with different colors for keywords, comments, and punctuation.

```
using System;
using System.Windows.Forms;

namespace myApp
{
    class Program
    {
        static void Main(string[] args)
        {
            Day myday = Day.Sat;
            if (myday == Day.Sat) {
                MyClass Goodluck=new MyClass();
                Goodluck.Myfunc("Hanwen");
            }
            Console.WriteLine("Hello World!");
        }
        enum Day { Sat, Sun };
    }
    class MyClass
    {
        public void Myfunc(string name)
        {
            MessageBox.Show("Say hi to " + name + "!");
        }
    }
}
```

IntelliSense – Coding Sense

你的第二大脑

The screenshot shows a dark-themed code editor window titled "Untitled-1". The code is written in C# and defines a class `Program` with a `Main` method and an `enum Day`. It also defines a class `MyClass` with a `Myfunc` method. The cursor is positioned at the end of the line `Goodluck.Myfunc("Hanwen" + i.)` in the `Main` method. A tooltip window displays several Intellisense suggestions:

- Equals()
- GetHashCode()
- GetType()
- ToString()

The "ToString()" suggestion is highlighted with a blue selection bar.

```
1 using System;
2 using System.Windows.Forms;
3
4 namespace myApp
5 {
6     class Program
7     {
8         static void Main(string[] args)
9         {
10             int i = 0;
11             Day myday = Day.Sat;
12             if (myday == Day.Sat)
13             {
14                 MyClass Goodluck = new MyClass();
15                 Goodluck.Myfunc("Hanwen" + i.);
16             }
17             Console.WriteLine("Hello World!");
18         }
19         enum Day { Sat, Sun };
20     }
21     class MyClass
22     {
23         public void Myfunc(string name)
24         {
25             MessageBox.Show("Say hi to " + name + "!");
26         }
27     }
28 }
```

At the bottom of the editor, there are status indicators: 0 errors, 0 warnings, 15 spaces used, UTF-8 encoding, CRLF line endings, and C# as the current language.

IntelliSense – Color Sense

世界不能失去色彩

The screenshot shows a dark-themed IDE interface with a menu bar (File, Edit, View, Debug, IntelliSense, Terminal, Help) and a title bar (*Untitled-1). The main area displays a C# program with color-coded syntax:

```
1  using System;
2  using System.Windows.Forms;
3
4  namespace myApp
5  {
6      class Program
7      {
8          static void Main(string[] args)
9          {
10             int i = 0;
11             Day myday = Day.Sat;
12             if (myday == Day.Sat)
13             {
14                 MyClass Goodluck = new MyClass();
15                 Goodluck.Myfunc("Hanwen" + i.ToString());
16             }
17             Console.WriteLine("Hello World!");
18         }
19         enum Day { Sat, Sun };
20     }
21     class MyClass
22     {
23         public void Myfunc(string name)
24         {
25             MessageBox.Show("Say hi to " + name + "!");
26         }
27     }
28 }
```

The code uses color coding for different elements: blue for `using`, `namespace`, `class`, `static`, `int`, `Day`, `if`, `MessageBox`, and `Console.WriteLine` methods; green for `Myfunc` and `ToString`; orange for strings; and purple for the `enum` keyword.

The status bar at the bottom shows: 0 errors, 0 warnings, 15 spaces used, UTF-8 encoding, CRLF line endings, and C# as the language.

IntelliSense – File Sense

数据无价，不再惧怕突然断电

The screenshot shows a code editor interface with the following details:

- Menu Bar:** File, Edit, View, Debug, **IntelliSense**, Terminal, Help.
- IntelliSense Submenu:** Structure Sense, Coding Sense, Color Sense, **File Sense**.
- Code Editor:** Untitled-1 window containing C# code.

```
using System;
namespace MyNamespace
{
    class Program
    {
        static void Main(string[] args)
        {
            int i = 0;
            Day myday = Day.Sat;
            if (myday == Day.Sat)
            {
                MyClass Goodluck = new MyClass();
                Goodluck.Myfunc("Hanwen" + i.ToString());
            }
            Console.WriteLine("Hello World!");
        }
        enum Day { Sat, Sun };
    }
    class MyClass
    {
        public void Myfunc(string name)
        {
            MessageBox.Show("Say hi to " + name + "!");
        }
    }
}
```
- Status Bar:** Shows 0 errors, 0 warnings, 15 spaces, UTF-8 encoding, CRLF line endings, and C# language.