

**软件学院**

**《软件构造》解析器报告**

**设计题目： C#--B语言解析器**

**专业班级： 软件工程16-2班**

**指导老师： 朱少民**

2018年12月

目 录

[一、**语言介绍** 3](#_Toc329534661)

[1.1 用户需求说明 5](#_Toc329534664)

[1.2 系统需求说明 6](#_Toc329534667)

[二、系统规划 4](#_Toc329534662)

[**三、需求分析** 5](#_Toc329534663)

[3.1 用户需求说明 5](#_Toc329534664)

[3.1.1 数据需求 5](#_Toc329534665)

[3.1.2 事务需求 5](#_Toc329534666)

[3.2 系统需求说明 6](#_Toc329534667)

[四、数据库逻辑设计 7](#_Toc329534668)

[4.1 ER图 7](#_Toc329534669)

[4.2 数据字典 8](#_Toc329534670)

[4.3 关系表 1](#_Toc329534671)0

[五、数据库物理设计 1](#_Toc329534672)1

[5.1 索引，视图，触发器，存储过程 1](#_Toc329534673)1

[5.2 安全机制 1](#_Toc329534675)5

[六、应用程序设计 1](#_Toc329534676)6

[6.1 功能模块 1](#_Toc329534677)6

[6.2 界面设计 1](#_Toc329534678)7

[七、总结 25](#_Toc329534681)

[附. 参考文献 2](#_Toc329534682)6

1. **语言介绍**

**1.1什么是B语言**

B语言是一种无类型语言，所有的数据都和机器字相对应，并且程序员可以直接通过内存地址操作内存。B语言最初在DEC的PDP-7和PDP-11小型机的早期Unix版本和Honeywell机器的GCOS系统上实现。

B语言的无类型特性使得其在PDP-11机器上无法以比较优雅的方式访问字符类型，因而从1971年开始，Dennis Ritchie开始对B语言进行改造，其中最重要的是引入了数据类型的概念，这些改造最终使B语言演化为了计算机发展历史上最为重要的编程语言——C语言。

时至今日，除了在一些古老的大型机或是特殊的嵌入式平台上，B语言已经芳踪难觅了。但B语言在编程语言发展历史上重要的承前启后的里程碑地位，却将永远为人们铭记。

**1.2 历史**

B语言是从[BCPL](https://baike.baidu.com/item/BCPL)系统中删减了汤普逊认为非必备的组件以便能运行在当时的[小型计算机](https://baike.baidu.com/item/%E5%B0%8F%E5%9E%8B%E8%AE%A1%E7%AE%97%E6%9C%BA)上而产生的。B语言还包括了汤普逊的一些个人偏好（主要在一些特定的程序中减少非空格[字符](https://baike.baidu.com/item/%E5%AD%97%E7%AC%A6)的数量）。

和BCPL以及FORTH类似，B语言只有一种数据类型，计算机字。大部分操作将其作为[整数](https://baike.baidu.com/item/%E6%95%B4%E6%95%B0)对待（例如进行+、-、\*、/操作），但其余操作将其作为一个复引用的[内存地址](https://baike.baidu.com/item/%E5%86%85%E5%AD%98%E5%9C%B0%E5%9D%80)。在许多方面B语言更像是一个早期版本的[C语言](https://baike.baidu.com/item/C%E8%AF%AD%E8%A8%80)，它还包括了一些[库函数](https://baike.baidu.com/item/%E5%BA%93%E5%87%BD%E6%95%B0)，其作用类似于C语言中的标准输入/输出函数库。

**1.3 示例**

下面是来自Ken Thompson的B语言用户手册的例子：

/\* The following function will print a non-negative number, n, to

the base b, where 2<=b<=10, This routine uses the fact that in the ANSCII character set, the digits 0 to 9 have sequential code values. \*/

printn(n,b) {

[extrn](https://baike.baidu.com/item/extrn) putchar;

auto a;

if(a=n/b) /\* assignment, not test for equality \*/

printn(a, b); /\* recursive \*/

putchar(n%b + '0');}

1. **开发环境**

所选语言：C#

运行环境：VS2013

运行系统：Windows10

引入类：Beval

项目名称：BLanguageParser

可执行程序：B语言解析器

1. **项目特色**

**3.1项目特点**

1. 实现了B 语言大部分关键字功能
2. B语言的声明变量时无需声明变量类型，在赋值时自动判定变量的类型
3. 实现了B语言数组的基本处理
4. 良好的可视化界面，可浏览、拖拽执行文件，文件内容会同步显示在左边的文本框内
5. 本项目一大难点在于opreator函数所处理的内容十分复杂，仅仅一个操作符就要应对多种变量形式（字符串型，数值型，数组型），而opreator函数要处理的是所有的操作符号（四则运算，逻辑运算等），在此，调用了网上开源的api减少了一部分工作量。
6. 另一大难点在于if while for语句等之间的复杂嵌套，需要缜密的逻辑才编写的完善正确。
7. 计算并显示了编译时间

**3.2运用知识点**

**3.2.1模块化设计**

项目分为2个自定义变量类，2个解析类，4个异常类及1个界面类，实现模块化设计，降低了耦合度，提高了可扩展性。自定义变量类中的变量设为私有，并设置了属性，供外部访问，对类进行了封装。4个异常类继承了Exception父类，简化了设计，区别不同的错误。

**3.2.2防御式编程**

1.对文件进行了过滤，使用户只能选择后缀为.txt的文件。

2.C#中断言的使用：

br = new System.IO.StreamReader(txt);

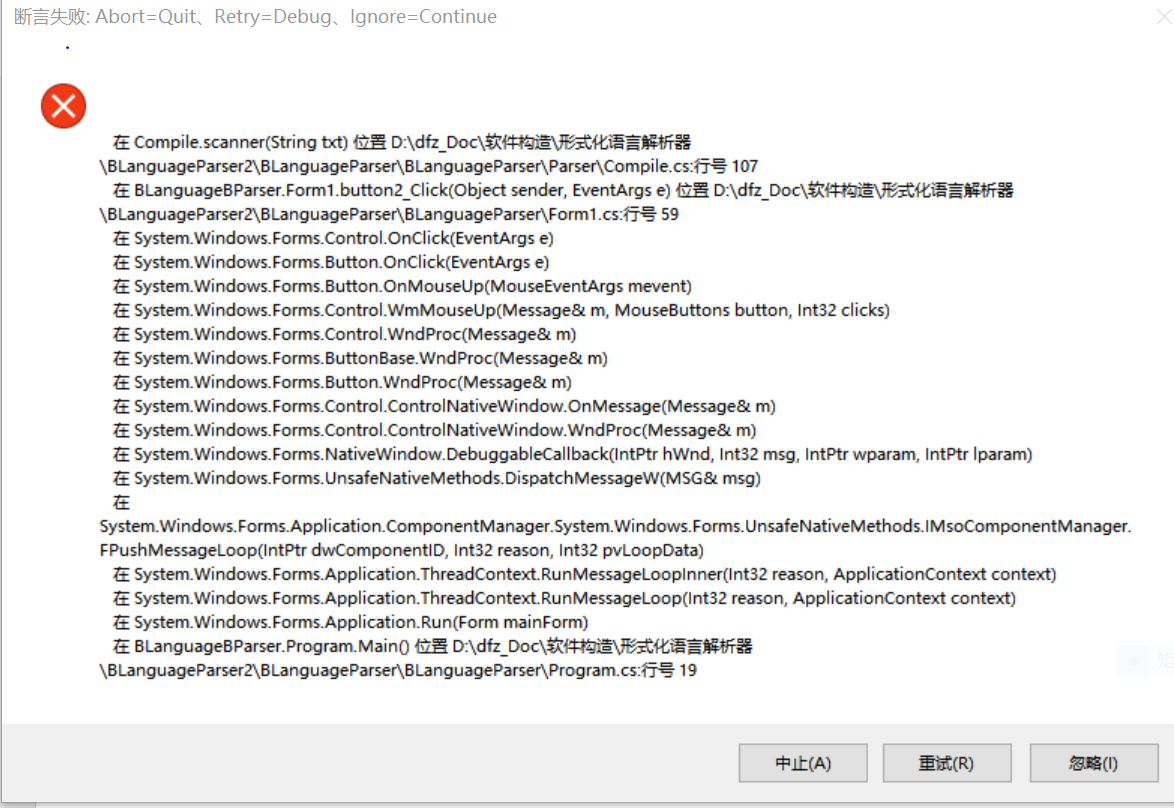
bool isEmpty = false;

if (br.ReadLine()==null)

isEmpty = true; //判断打开的.txt文件是否为空

Trace.Assert(!isEmpty); //如果打开文件为空，则触发断言.

说明：解析器在读取.txt文件时，如果文件内容为空，则触发断言。如下图：

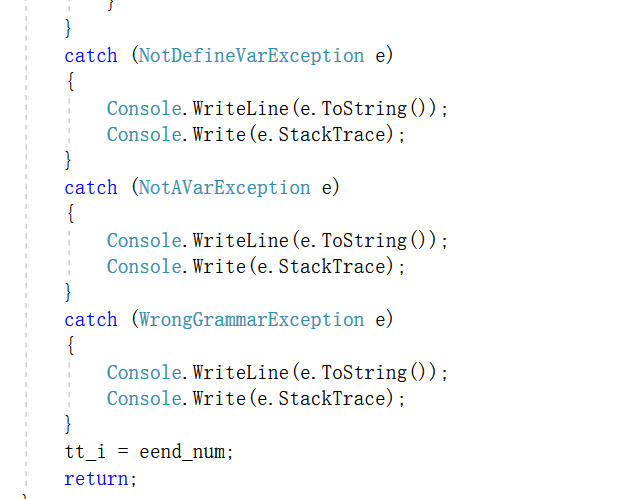


3.错误处理和异常机制

利用C#中的throw关键字将异常抛出



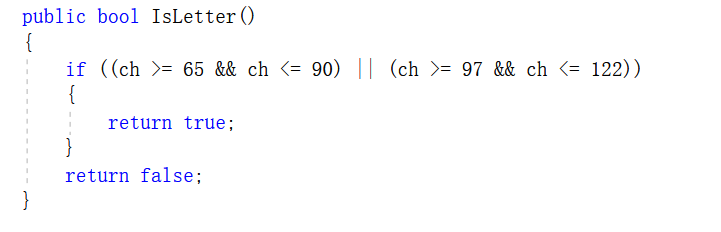
利用c#中的try…catch机制捕获并处理异常



**3.2.3变量的使用**

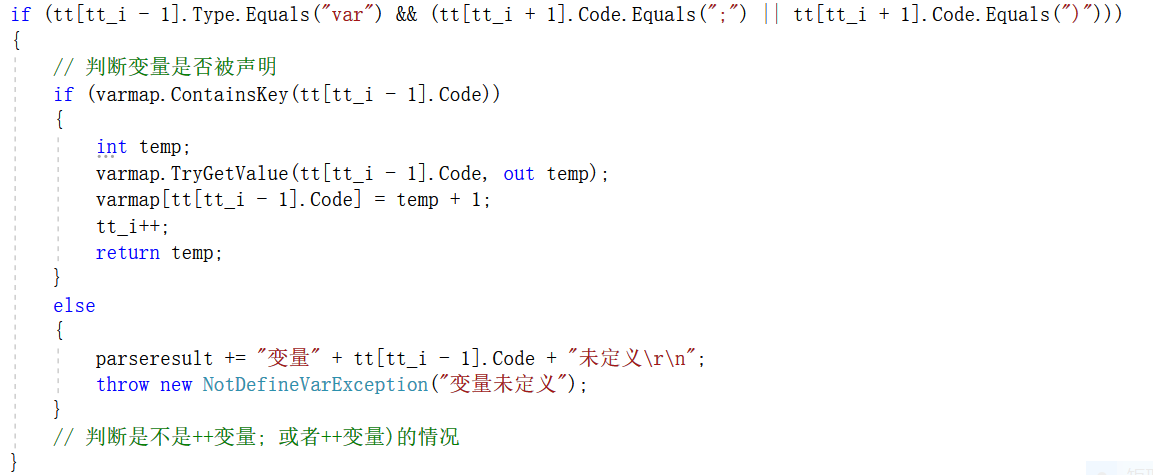
1.命名规范

公有函数名采用大驼峰命名方法，且做到见名知意：



2.在使用变量时，才声明并初始化变量

代码段中的temp变量，只有在使用时才声明并初始化，缩短变量的存活时间，尽量使变量的作用域最小化，生命周期最短：

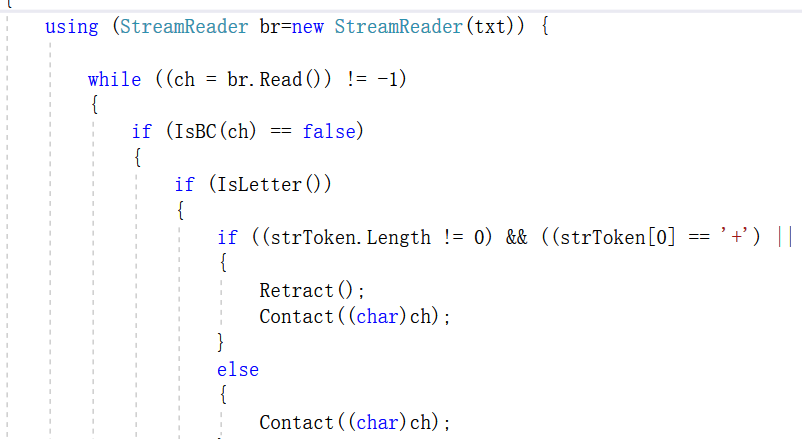


**3.2.4代码重构和优化**

1.代码优化：

使用.net中的using进行资源的管理：

下图中，using使用资源对象：文件读取流StreamReader，该对象实现了IDisposable接口.此接口提供了Dispose方法，该方法将释放此对象的资源，即在程序运行到using范围外时，主动释放文件读取流，释放资源。



2.代码重构：

1. **项目介绍**

**4.1 实现关键字(区分大小写)**

1. **auto**:变量名由字母混合数字组成，或纯字母组成，不能含空格等其他符号，数字不能用作开头。

B语言只有一种机器字数字类型，由auto定义变量，在赋值时确定数据类型。本解析器定义了数值和字符串两种数据类型，两种初始化变量方式如下：

数值型（默认初始化值为0）

1. auto 变量名 = 数值;
2. auto 变量名;

变量名 = 数值;

字符串型（默认初始化值为null）

1. auto 变量名 = ‘字符串’ ;
2. auto 变量名;

变量名 = ‘字符串’ ;

注:（‘=’两边必须有空格）

支持同时初始化多个变量 auto a , b;

1. **putnumb**:输出数值型变量或输出数值

putnumb(变量)； 输出变量的值

putnumb(数值)； 直接输出数值

1. **putchar**:输出字符串型变量或输出字符串

putchar(变量)； 输出变量的值

putchar(‘字符串’)； 直接输出字符串（不含空格）

1. **getchar**：输入字符串型变量值

getchar(变量名);

1. **getnumb**:输入数值型变量值

getnumb(变量名);

1. **goto**:

Goto 标号

程序跳转到指定标号之后开始执行

示例：

goto A;

putnumb (7);

A:

putchar('zpj');

程序打印zpj，跳过第二句不打印数字7

1. **if else**:

判断分支语句

If(A<B) {

\*\*\*\*\*

\*\*\*\*\*

} else {

\*\*\*\*\*

\*\*\*\*\*

}

1. **while**：

循环语句

auto i = 3;

while(i) {

i = i - 1;

putnumb(i);

}

输出2 1 0

**4.2 B语言测试示例**

求10个数的最大数和平均数：

auto a[10];

auto i , j;

putchar('Please enter 10 numbers');

i = 0;

A:

while(i < 10){

getnumb(j);

a[i] = j;

i++;

}

putchar('The 10 numbers are');

putstr(a); /\* 输出数组 \*/

putchar('The 10 numbers are');

auto m;

for(m = 0; m < 10; m++) {

putnumb(a[m]);

}

putchar('The largest of the ten numbers is');

auto k = 0;

auto max;

auto temp, sum;

max = a[0];

sum = 0;

while( k < 10 ){

temp = a[k];

if(temp > max) {

max = temp;

}

sum = sum + temp;

k++;

}

sum = sum / 10;

putnumb(max);

putchar('The average of 10 numbers is');

putnumb(sum);

**4.3 项目分析**

程序有2个自定义类型变量MyArray和WordType，4个异常处理类，和2个核心处理类词法分析器类Compile和解析器类Parser**，**选择一个文件后，经过Compile中的scanner函数处理成<Type, code>序列的形式保存在WordType类中，Parser解析器类中的BParser函数对这个序列进行分析，触发调用不同的函数，从而解析出正真运行的结果。

**1 Compile类中的scanner函数**

词法分析B语言，以字符流方式读取，判断读到的是否是关键字，+ - \* 、等操作符，变量名，goto的标签等，保存为<type,code>序列，即放入一个静态的WordType数组中，对应WordType类的两个属性。

注：在Compile类中初始化一个静态的<string,int>markmap用于保存读到的所有goto语句的标签，以及他们对应的序列下标。

**2 Parser类以及Bparse的实现**

首先初始化一个Compile类，用来获取词法分析器分析出的WordType序列, 接着初始化<string, string>和<string , int>两个varmap用于存储auto声明的变量，以及他们对应的value，由于只有在初始化时才能确定变量的类型，所以需要同时给一个变量存有两种形态。

Bparser函数中只有一个while循环，遍历词法分析器解析的WordType序列，判断对应的Type跳转到对应的函数去处理。

**3 auto函数**

BParser函数读取到auto关键字，跳转到auto函数，在函数内部，首先判断下一位的类型是不是变量型（var），如果不是，则报错，如果是，就将变量对应的名称和初始值（0和null）存入两个map里，接着判断变量后面一位是不是等于号，如果是等于号就跳转到operator函数对变量进行赋值操作后返回；如果下一位是逗号就进行递归操作（调用auto函数本身）；如果变量的下一位是分号，则直接让tt\_i自增并返回。

**4 putnumb函数**

BParser函数读取到putnumb关键字，跳转到putnumb函数，在函数内部，首先判断下一位的code是否是左括号，不是就报错，如果是就将下标加一，接着判断接下来一位的类型是否是变量并且下下一位是否是右括号，并且下下下一位是否是分号，如果是则判断该变量是否在数值型markmap表中，若在则输出该变量对应的value并返回下标加2，如果不满足接下来一位的类型是否是变量并且下下一位是否是右括号，并且下下下一位是否是分号，则判断下一位的类型是否是数值常量并且下下一位是右括号，并且下下下一位是分号，如果是则打印数值并返回下标加2，如果都不满足就报错

**4 putchar函数**

原理同putnumb函数相同，只是在打印字符串时要判断字符串前后的单引号。

而后加了判断，当输出的是用单引号包围的字符串时，进行循环判断输出两个单引号之间的所内容，并且在读取到\*n时执行换行输出操作。

**5 getnumb函数**

BParser函数读取到getnumb关键字，跳转到getnumb函数，在函数内部，首先判断下一位是否是左括号，并且下下一位的类型是变量名，并且下下下一位是右括号，并且下下下下一位是分号，满足这些条件说明语法通过，不满足则不报错。接下来判断变量是否已经声明过，通过判断该变量是否在markmap表中。如果存在，new一个scanner对象，把从控制台输入的值赋给数值型markmap表中key值为该变量名对应的value。处理下标，返回到分号位置对应的下标号。

**6 getchar函数**

原理同getnumb函数相同 只是把从控制台输入的值赋给字符串型markmap表中key值为该变量名对应的value。。

**7 Bgoto函数**

BParser函数读取到goto关键字，跳转到Bgoto函数，在函数内部，首先判断下一位的类型是否是goto的跳转标签（mark），如果不是则报错，如果是就从标签对应的markmap中找到该标签对应的value值，也就是标签之后的序列下标，将其赋值给当前下标tt\_i 并return出去。

**8 Bopreator函数**

当operator函数被调用时传入当前的正在处理的操作符，对操作符进行判断，分为两类，一类是“=”等于运算，另一类是“+”、“-”……这些关系运行，第一类对应的是赋值运行，第二类对应的是逻辑判断。

在处理第一类问题时，需要判断等号左边的变量类型是普通变量还是数组变量，之后临时存储变量的位置，等待等号右边处理完后赋值给该变量。等号右边一直读取到“；”结束符，然后将所有关键字转换成一个string字符串，如果等号右边是字符串直接赋值给等号左边的变量，如果是算式运算则调用Beval直接处理算式运算，运算结束之后转换数据类型赋值给等号左边的变量，等号运算结束，返回默认值1表示运算成功。

在处理第二类问题时，需要从运算符左边一直读取到结束符“){”结束，读取一个完整的逻辑运算表达式，中间需要将变量和数组全部转换成相应的数据并追加进入字符串中，然后调用Beval直接运算出结果，数据类型转成int返回该值。

**9 Bwhile函数**

当BParser读取到while关键字时，跳转到对应的BWhile函数执行，首先循环判断while语句是否正确：当到左花括号时将对应参数加一，当读到右花括号时将上述的参数减一，当参数为零时停止循环；而后利用C#的while函数来执行：首先调用Wcheck函数检查判断条件是否成立，如果成立则进入C#对应的while函数，将while语句的起始位置传递给Parser函数执行。

**10 Bif函数**

当BParser读取到if关键字时，跳转到Bif函数执行，首先循环判断if语句是否正确，而后检查是否有else语句，如果没有就将if语句的结尾标志赋值为紧接着if的右花括号所对应的位置的下标，如果有else语句就将对应的是否具有else语句的标志置为true并且将else语句结尾标志赋值为紧接着else的右花括号所对应的位置的下标。而后进行判断执行，具体介绍参看类里的注释。

**11 unaryOperation函数**

分为读到++和--两种情况，在++和--中又分为变量在前和变量在后两种情况，例如变量在前，首先判断语法是否正确（满足 变量++； 或者 ++变量；）如果满足则判断变量是否已经被声明，若被声明，则将该变量的值加一，将tt\_i右移至分号所在处，然后返回该变量原来的值，若变量在后，则返回更新后的变量值。

**12 BDo\_while函数**

BDo\_while函数在读取到do关键字后，跳转到BDo\_while方法，在判断语法没有问题后继续，分别将do语句判断与执行开始和结束处的下标赋给两个局部变量，再判断是否存在while语句，若存在则将判断语句开始和结束处的下标赋给另外两个局部变量，先将do执行语句下标为参数调用Parser方法再将判断语句下标作为参数调用WCheck方法，若正确继续调用do执行语句,若错误，则读取下一个字。

**13 Bfor函数**

BParser函数在读取到for关键字后，跳转到Bfor方法，先将圆括号中的语句进行扫描，计算分号数量，从而进行语法的判断，若没有错误，以分号为分隔符将圆括号中的语句分为三个部分，将各自的语句开始和结束处的下标赋分别给两个局部变量，并将右圆括号改为分号，保证语句的完整性。然后将循环语句的开始和结束的下标赋给两个局部变量。首先以第一部分的下标为参数调用FParser，然后使用while语句以第二部分的下标为参数调用WCheck进行判断若为假将全局变量指向循环结束下标，若正确进行以第三部分下标和循环部分下标为参数的FParser调用达到循环的效果。

**14 Note函数**

处理思路，函数读取到note关键字跳转到对应函数执行，note函数读取到的类型为note时进行判断，如果是左注释就将是否有左注释的标志置为真，在读取到右注释时判断是否存在左注释，如果不存在就报错，如果存在就将全局下标tt\_i指向右注释所在位置，如果读取到函数体最后还是未读取到右注释则报错。

**15 Putstr函数**

首先判断语法逻辑，如果满足putstr（变量）；的形式，然后判断变量是否是已经声明过的数组，如果不是就抛出异常，如果是，则循环输出该数组

**4.4 项目示例**

1.最大数和平均数

auto a[10];

auto i , j;

putchar('Please enter 10 numbers');

i = 0;

A:

while(i < 10){

getnumb(j);

a[i] = j;

i++;

}

putchar('The 10 numbers are');

putstr(a); /\* 输出数组 \*/

putchar('The 10 numbers are');

auto m;

for(m = 0; m < 10; m++) {

putnumb(a[m]);

}

putchar('The largest of the ten numbers is');

auto k = 0;

auto max;

auto temp, sum;

max = a[0];

sum = 0;

while( k < 10 ){

temp = a[k];

if(temp > max) {

max = temp;

}

sum = sum + temp;

k++;

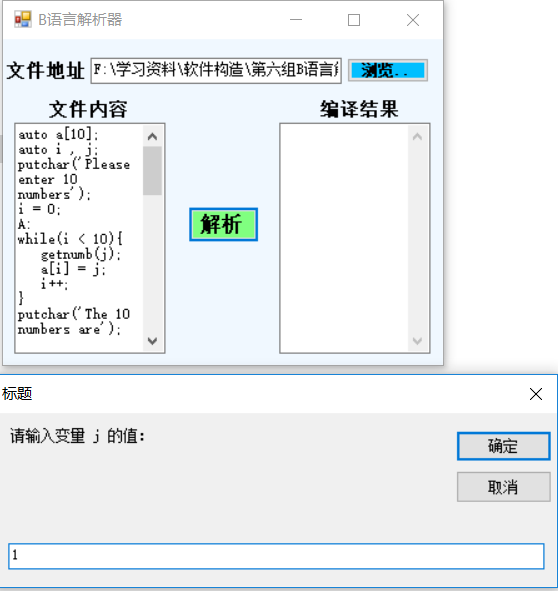
}

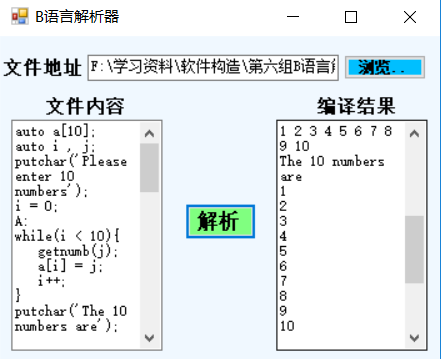
sum = sum / 10;

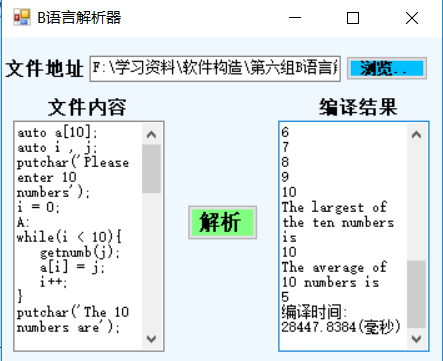
putnumb(max);

putchar('The average of 10 numbers is');

putnumb(sum);







2.do\_while

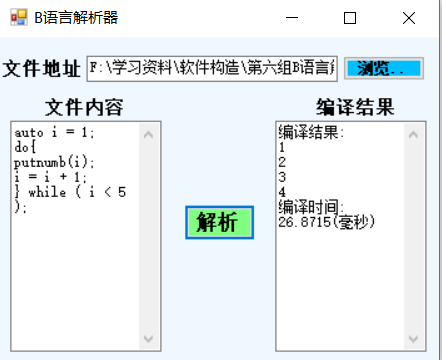
auto i = 1;

do{

putnumb(i);

i = i + 1;

} while ( i < 5 );



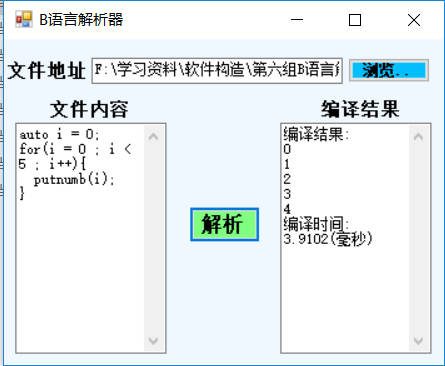
3.For

auto i = 0;

for(i = 0 ; i < 5 ; i++){

putnumb(i);

}



4.Goto

auto i;

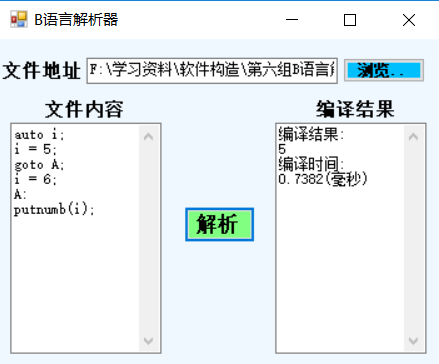
i = 5;

goto A;

i = 6;

A:

putnumb(i);



5.If

auto i = 5;

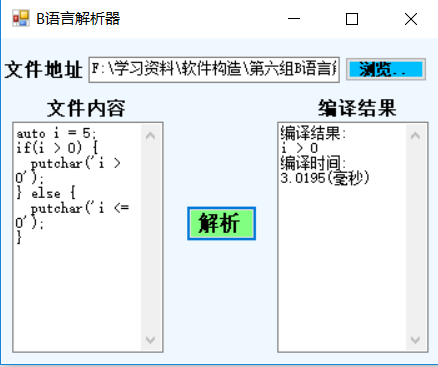
if(i > 0) {

putchar('i > 0');

} else {

putchar('i <= 0');

}



6.While

auto i = 5;

while(i){

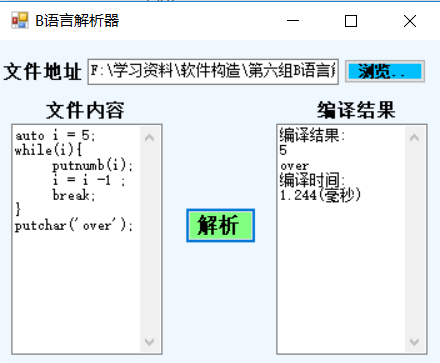
putnumb(i);

i = i -1 ;

break;

}

putchar('over');



7.while嵌套if

auto i = 0;

while(i < 3){

putnumb(i);

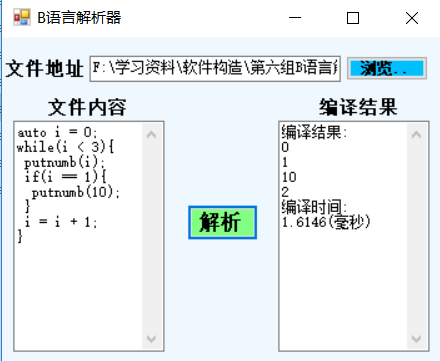
if(i == 1){

putnumb(10);

}

i = i + 1;

}



8.变量初始化及输出

auto a, b;

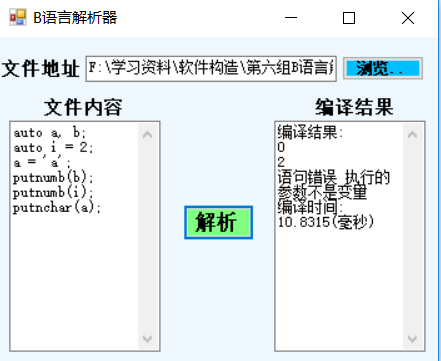
auto i = 2;

a = 'a';

putnumb(b);

putnumb(i);

putnchar(a);



9.输入

auto a, b, c;

getchar(a);

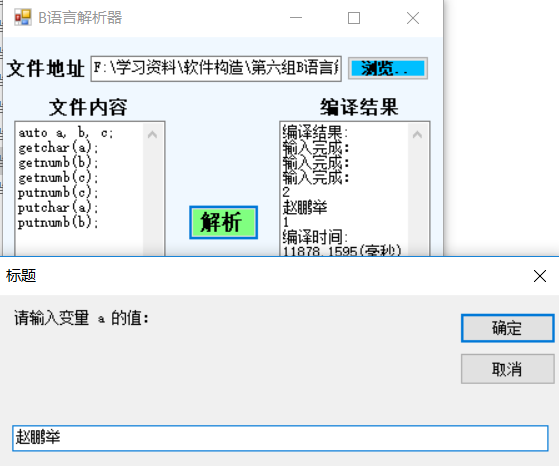
getnumb(b);

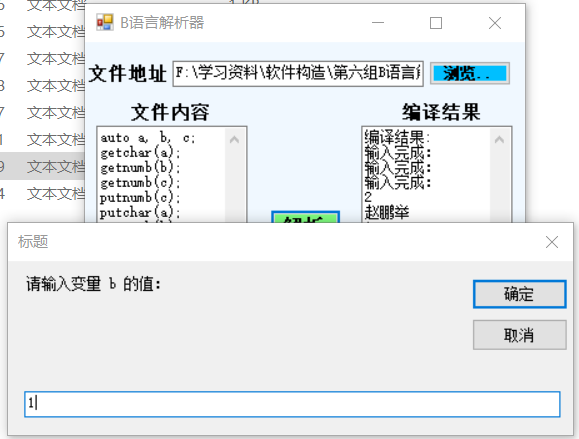
getnumb(c);

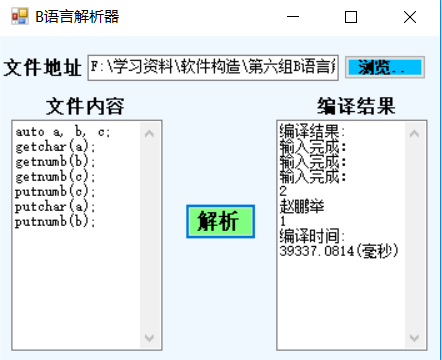
putnumb(c);

putchar(a);

putnumb(b);







# 附. 参考文献

**参考文献：**

《代码大全》

《编译原理》

《C#图解教程》

**参考网站：**

<https://www.bell-labs.com/usr/dmr/www/btut.html> B语言学习网址<http://blog.csdn.net/u012577528/article/details/45315925> 词法分析器参考网址