《编译技术》课程设计
文　　档

学号：_____16182678_____

姓名：_____李继诗_____

2019 年　1　月　4　日

目录

一．需求说明

1．文法说明

难度系数 2，**扩充** C0 文法；
文法中**不含**需要消除的左递归，**未**对文法进行改写和扩充；
文法具体说明详见**文法分析**作业；

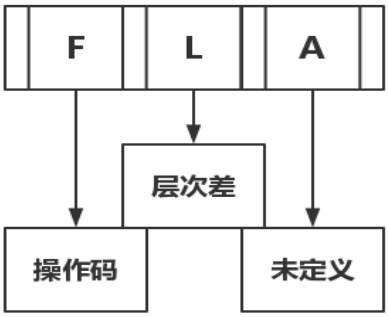
```
<加法运算符> ::= + | -
<乘法运算符> ::= * | /
<关系运算符> ::=  < | <= | > | >= | != | ==
<字母>    ::= _ | a | . . . | z | A | . . . | Z
<数字>    ::= 0 | <非零数字>
<非零数字>  ::= 1 | . . . | 9
<字符>     ::=  '<加法运算符>' | '<乘法运算符>' | '<字母>' | '<数字>'
<字符串>    ::=  "{十进制编码为 32,33,35-126 的 ASCII 字符}"
<程序>     ::= [<常量说明>] [<变量说明>]
               {<有返回值函数定义>|<无返回值函数定义>}<主函数>
<常量说明>   ::=  const<常量定义>;{ const<常量定义>;}
<常量定义>   ::=   int<标识符>=<整数>{,<标识符>=<整数>}
                 | float<标识符>=<实数>{,<标识符>=<实数>}
                 | char<标识符>=<字符>{,<标识符>=<字符>}
<无符号整数>  ::=  <非零数字>{<数字>} | 0
<整数>     ::= [+ | -]<无符号整数>
<小数部分>   ::=  <数字>{<数字>}
<实数>     ::=  <整数>.<小数部分>
<标识符>    ::=  <字母>{<字母> | <数字>}
<声明头部>   ::=  int<标识符> |float <标识符>|char<标识符>
<变量说明>   ::=  <变量定义>;{<变量定义>;}
<变量定义>   ::=  <类型标识符>(<标识符>|<标识符>'['<无符号整数>']')
                 {,(<标识符>|<标识符>'['<无符号整数>']')}
                 //<无符号整数>表示数组元素个数，其值需大于 0
<可枚举常量>  ::=  <整数>| <字符>
<类型标识符>  ::=  int | float | char
<有返回值函数定义>  ::=  <声明头部>'('<参数表>')' '{'<复合语句>'}'
<无返回值函数定义>  ::=  void<标识符>'('<参数表>')''{'<复合语句>'}'
<复合语句>   ::= [<常量说明>] [<变量说明>] <语句列>
<参数表>    ::=  <参数>{,<参数>}|<空>
<参数>     ::=  <类型标识符><标识符>
<主函数>    ::=  void main'('')' '{'<复合语句>'}'
<表达式>    ::= [+ | -] <项>{<加法运算符><项>}
                // [+|-]只作用于第一个<项>
<项>      ::=  <因子>{<乘法运算符><因子>}
<因子>     ::=  <标识符> | <标识符>'['<表达式>']' | <整数>|<实数>
               |<字符> | <有返回值函数调用语句>|'('<表达式>')'
<语句>     ::=  <条件语句> | <循环语句>| <有返回值函数调用语句>;
               | <无返回值函数调用语句>; | <赋值语句>; | <读语句>;
               | <写语句>; | <空>;|<情况语句> | <返回语句>;
<赋值语句>   ::=  <标识符>=<表达式>|<标识符>'['<表达式>']'=<表达式>
<条件语句>   ::=  if '('<条件>')'<语句>[else<语句>]
<条件>     ::=  <表达式><关系运算符><表达式> | <表达式>
               //表达式为 0 条件为假，否则为真
<循环语句>   ::=  while '('<条件>')'<语句>
<情况语句>   ::=  switch '('<表达式>')' '{'<情况表>[<缺省>]'}'
<情况表>    ::=  <情况子语句>{<情况子语句>}
<情况子语句>  ::=  case<可枚举常量>：<语句>
<缺省>     ::=  default : <语句>
<有返回值函数调用语句>  ::= <标识符>'('<值参数表>')'
<无返回值函数调用语句>  ::= <标识符>'('<值参数表>')'
<值参数表>   ::= <表达式>{,<表达式>} | <空>
<语句列>    ::=  {<语句>}
<读语句>    ::=  scanf '('<标识符>{,<标识符>}')'
<写语句>    ::=  printf '('  <字符串>,<表达式>  ')'
               | printf '('<字符串>  ')'|printf '('<表达式>')'
<返回语句>   ::=  return['('<表达式>')']
```

（文法展示）

2. 目标代码说明

• 类 P-code 语言

• 指令格式

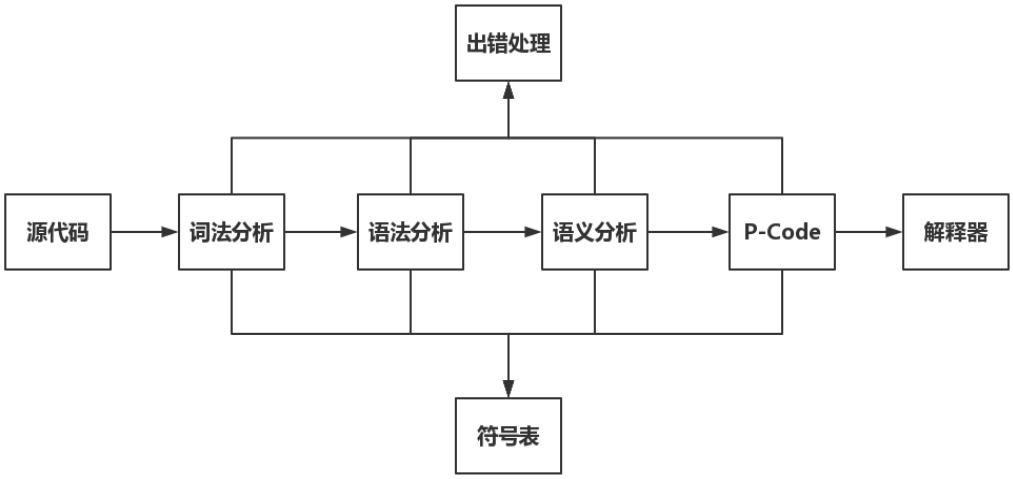


- **操作码：**指令助记符，例：LOD、STO 等；
- **层次差：**表示引用层与声明层之间的层次差，默认为 0；
- **未定义（A）：**不同的指令含义不同，可以是基地址，也可以储存值等等；

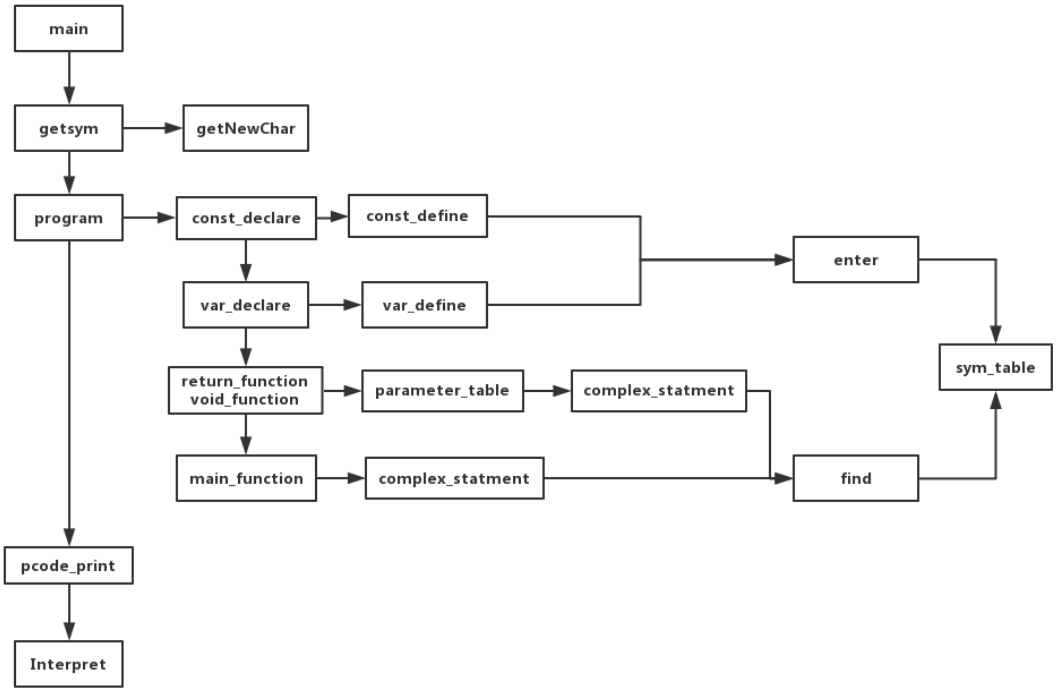
类 P-code 语言常用指令集				
助记符	F	L	A	功能
LDI	0	X	Y	装入立即数
LDA	1	X	Y	把变量地址装入栈顶
LOD	2	X	Y	装入值
IXX	3	X	Y	取下标地址
STO	8		Y	将栈顶内容存入以栈顶次高元为地址的单元
JPC	9		Y	如栈顶内容为假，转移到 Y
JMP	10		Y	无条件转移到 Y
RED	11		Y	读操作
WRR	12		Y	写操作
ADD	13		Y	加操作
SUB	14		Y	减操作
MUL	15		Y	乘操作
DIV	16		Y	除操作
MUS	17		Y	求负操作
GRT	18		Y	大于
LES	19		Y	小于
EQL	20		Y	等于
NEQ	21		Y	不等于
GEQ	22		Y	大于等于
LER	23		Y	小于等于
CAL	24		Y	函数调用
RET	25		Y	函数返回值
EXF	26		Y	函数调用结束
INF	27		Y	参数声明

二．详细设计

1．程序结构



（示意图）



（示意图）

- 以**语法分析**和**语义分析**为核心，调用**词法分析**作为子程序；
- **符号表**和**错误处理作**为辅助；
- **词法分析**按位读入源代码，返回**单词类别码**和**单词值**；
- 识别到**标识符**或**关键字**时，调用**符号表**子函数；
- 最终生成**类 P-code 语言**，并自己构建**类 P-code 解释器**；

2．类/方法/函数功能

- 词法分析

函数	功能
void getsym()	确认单词类型
void getNewchar()	获取下一字符

- 语法分析

函数	功能
void program()	程序
void const_declare()	常量声明
void var_declare()	变量声明
void return_function()	有返回值函数
void void_function()	无返回值函数
void main_function()	主函数
void const_define()	常量定义
void var_define()	变量定义
void parameter_table()	参数表
void complex_statment()	复合语句
void statment_list()	语句列
void statment()	语句
void ifstatment()	条件语句
void whilestatment()	循环语句
void casestatment()	情况语句
void scanfstatment()	读语句
void printfstatment()	写语句
void returnstatment()	返回语句
void assignatatment()	赋值语句
void returnfuncstatment()	有返回值函数调用语句
void voidfuncstatment()	无返回值函数调用语句
void condition()	条件
void case_table()	情况表
void defaultstatment()	缺省
void case_son()	情况子语句
void expression()	表达式
void term()	项
void factor()	因子
void value_parameter_table()	值参数表

- 符号表相关

函数	功能
void isFull()	检查符号表是否已满
void NewNode()	创建新节点
void CreateConstInt()	创建新整型常量
void FillConstInt()	完善整型常量
void CreateConstFloat()	创建新浮点常量
void FillConstFloat()	完善浮点常量
void CreateConstChar()	创建新字符常量
void FillConstChar()	完善字符常量
void CreateInt()	创建整型变量
void CreateIntArray()	创建整型数组
void CreateFloat()	创建浮点变量
void CreateFloatArray()	创建浮点数组
void CreateChar()	创建字符变量
void CreateCharArray()	创建字符数组
void CreateIntFun()	创建整型函数
void CreateFloatFun()	创建浮点函数
void CreateCharFun()	创建字符函数
void CreateVoidFun()	创建无返回值函数
void CreateMainFun()	创建主函数
void CreateIntPara()	创建整型参数
void CreateFloatPara()	创建浮点参数
void CreateCharPara()	创建字符参数
int isReDefine()	检查是否重定义
int Find(char name[])	查找符号表

- 代码生成

函数	功能
void gen_int(int op, int lev, int x)	产生整型表达式
void gen_float(int op, int lev, float x)	产生浮点表达式
void gen_char(int op, int lev, char x)	产生字符表达式
void pcode_print()	打印表达式

- 解释执行

函数	功能
void Interpret()	解释执行 Pcode

- 错误处理

函数	功能
void Error(int e)	错误处理

3．调用依赖关系

- 词法分析
  - 由 **getsym()**调用 **getNewchar()**子程序，从**源代码**中读出单词符号；
  - 在**语法分析**有需要时，调用**词法分析**；

- 语法分析
  - 采用**自顶向下**的**递归子程序**法；

  program()调用 **const/var_declare()、return/void/mainfunction()**；
  const/var_declare()调用 **const/var_define();**
  return/voidfunction()调用 **parameter_table()、complex_statment();**
  mainfunction()调用 **complex_statment();**
  complex_statment()调用 **statment_list();**
  statment_list()调用 **statement();**
  statement()调用 **if / while / casestatment()、scanf / printfstatment()、**
  returnstatment()、assignatatment()、
  return / voidfuncstatment();
  ifstatment()调用 **condition()、complex_statment();**
  while statment()调用 **condition()、complex_statment();**
  casestatment()调用 **case_table()、defaultstatment()、**
  case_son()、complex_statment();
  return/voidfuncstatment()调用 **value_parameter_table(),complex_statment();**
  statement()调用 **expression();**
  expression()调用 **term();**
  term()调用 **factor()**；

- 符号表相关
  - 调用 **isFull()**检查**符号表**是否已满
  - 调用 **find()**检查**符号表**中是否已经存在；
  - 调用 **NewNode()**在**符号表**中建立新节点；
  - 调用 **Create / Fill 函数**将**标识符/保留字**填入**符号表**；

- 代码生成
  - 调用 **gen_int / float / char()**生成 **Pcode 表达式;**
  - 调用 pcode_print()打印并储存 **Pcode 表达式;**

- 解释执行
  - 调用 Interpret()**解释执行 Pcode 表达式;**

- 错误处理
  - 调用 Error()进行**错误处理**，报错并跳过错误代码；

4．符号表管理方案

符号表设计详解			
名称	类型	描述	
name	char	存储标识符名称	
valid	int	有效位	
kind	int	存储标识符种类	0：常量
			1：变量
			2：数组
			3：有返回函数
			4：无返回函数
			5：主函数
			6：参数
type	int	存储标识符类型	0：整型
			1：浮点
			2：字符
value	int	存储整型数值	
value_float	float	存储浮点数值	
level	int	标识符所在层次	0：全局
			1：第一层
			……
line	int	存储标识符所在行号	
addr	int	存储数组首地址或函数跳转地址	

5．存储分配方案

（1）数据栈

- 首先通过声明，预留**全局变量**数据位；
- 运行时为每个**常量**和**变量**开辟数据位；
- **数组变量**通过**基地址**+**偏移量**确定位置；
- 函数退出后，**栈指针**回到初始位置；

（2）运行栈

- 执行运算操作时，从**数据栈**中加载数据，在此运算；
- 运算完毕后，按照指令，将数据写回或者输出，**栈指针**减；
- 理论上，一直是**栈顶**和**次栈顶**参与运算；

（3）指针栈

- 执行跳转时保存**指令栈指针**；
- 函数结束时令**指令栈指针**回到跳转前的状态；

6. 解释执行程序

- 首先，将 **Pcode** 文本读入，并记录在**指令栈**中；
- 确定 **main** 入口，并为**全局变量**在**数据栈**开辟存储空间；
- 移动**指令栈指针**，读入**当前指令**，将数据放入**数据栈**；
- 根据指令，在**运算栈**中运算数据；
- 根据指令，将数据**存入指定地址**或**输出到屏幕**；

7. 出错处理

- **词法错误**

编号	描述
101	叹号单独出现
102	缺少后引号
103	引号内非合法字符
104	单引号内仅限单字符
105	数字不得以 0 开头

- **语法错误**

编号	描述
201	缺少主函数
202	应以 const 开头
203	应以；结束
204	应为标识符
205	应为=
206	非整数
207	非无符号整数
208	非实数
209	非字符
210	非类型标识符
211	格式不合法
212	缺少后括号
213	应以 void 开头
214	非语句
215	应以 if 开头
216	应以 while 开头
217	应以 switch 开头
218	应以 case 开头
219	非可枚举常量

220	应为:
221	应以 default 开头
222	应以 scanf 开头
223	应以 printf 开头
224	应以 return 开头
225	常量不得赋值
226	实参个数不匹配
227	数组越界
228	赋值类型不匹配

- **符号表错误**

编号	描述
301	符号表已满
302	标识符重定义

- **代码生成错误**

编号	描述
401	标识符未定义
402	常量不许被定义

三．操作说明

1．运行环境

- 操作系统：**Windows 10**
- 开发语言：**C 语言**
- 开发环境：**Visual Studio 2017**

2．操作步骤

- 打开工程文件 **Compile.sln**；
- 选中源文件，**编译运行**；
- 在控制台中输入**源代码路径**；
- 生成目标代码 **P-Code /** 进入**错误处理**；
- **解释执行**目标代码 **P-Code ;**
- 在窗口进行**输入操作**；

四．测试报告

1．测试程序及测试结果

（1）test_fib.txt

```
1.  int test_fib(int n)
2.  {
3.        if (n == 0)  return (0);
4.        else  if (n == 1)  return (1);
5.        else   return(test_fib(n - 1) + test_fib(n - 2));
6.  }
7.  void main()
8.  {
9.        int fib;
10.       scanf(fib);
11.       fib = test_fib(fib);
12.       printf("fib = ", fib);
13. }
```

输入和输出					
输入	0	1	2	3	……
输出	fib = 0	fib = 1	fib = 1	fib = 2	……

（2）test_ while.txt

```
1.  void test_while(int test)
2.  {
3.          while (test > 0)
4.                  printf(test);
5.                  test = test - 1;
6.  }
7.
8.  void main()
9.  {
10.         int test;
11.         scanf(test);
12.         test_while(test);
13. }
```

输入和输出					
输入	1	2	3	4	……
输出	1	2 1	3 2 1	4 3 2 1	……

（3）test_ switch.txt

```
1.  void test_switch(int test)
2.  {
3.      switch (test)
4.      {
5.      case 1:  printf("CASE 1");
6.      case 2:  printf("CASE 2");
7.      case 3:  printf("CASE 3");
8.      default: printf("DEFAULT");
9.      }
10. }
11.
12. void main()
13. {
14.      int test;
15.      scanf(test);
16.      test_switch(test);
17. }
```

输入和输出					
输入	1	2	3	4	……
输出	CASE 1	CASE 2	CASE 3	DEFAULT	……

（4）test_ max.txt

```
1.  int max(int a, int b, int c)
2.  {
3.         if(a >= b)  d = a;
4.         else d = b;
5.         if(d <= c) d = c;
6.  }
7.
8.  void main()
9.  {
10.         int a, b, c, d;
11.         scanf(a, b, c);
12.         max(a, b, c);
13.         printf("max is ", d);
14. }
```

输入和输出					
输入	0 1 2	0 2 1	2 1 0	1 1 0	……
输出	max is 2	max is 2	max is 2	max is 1	……

（5）test_ compute.txt

```
1.  const int a = 1;
2.  const float b = 2.0;
3.  const char c = 'c';
4.
5.  void main()
6.  {
7.          int x, y, z;
8.          int array[10];
9.          scanf(x, y, z);
10.         array[5] = z;
11.         x = x / 1;
12.         y = y * b;
13.         array[5] = array[5] - c;
14.         printf(x, y, array[5]);
15. }
```

输入和输出					
输入	0 1 2	1 2 3	2 3 4	3 4 5	……
输出	0 2 97	1 4 96	2 6 95	3 8 94	……

（6）error1.txt

```
1.  const int a = 1;
2.  const float b = 2.0;
3.  const char c = 'c';
4.
5.  void main()
6.  {
7.          int x, y, z;
8.          int  array[10];
9.          scanf(x, y, z);
10.         array[11] = z;
11.         x = x / 1;
12.         y = y * b;
13.         array[11] = array[11] - c;
14.         printf(x, y, array[11]);
15. }
```

错误信息
Line 10/13/14：数组越界

（7）error2.txt

```
1.  const int a = 1;
2.  const float b = 2.0;
3.  const char c = 'c';
4.
5.  void main()
6.  {
7.          int x, y, z;
8.          int  array[10];
9.          scanf(x, y, z)
10.         array[5] = z;
11.         x = x / 1;
12.         y = y * b;
13.         array[5] = array[5] - c;
14.         printf(x, y, array[5]);
15. }
```

错误信息
Line 9：应以；结束

（8）error3.txt

```
1.  const int a = 1;
2.  const float b = 2.0;
3.  const char c = 'c';
4.
5.  void main()
6.  {
7.          int x, y, z;
8.          int  array[10];
9.          scanf(x, y, z);
10.         array[5] = z;
11.         x = x / a;
12.         a = y * b;
13.         array[5] = array[5] - c;
14.         printf(x, y, array[5]);
15. }
```

错误信息
Line 12：常量不得赋值

（9）error4.txt

```
1.  const int a = 1;
2.  const float b = 2.0;
3.  const char c = 'c';
4.
5.  void main()
6.  {
7.          int x, y, z;
8.          int  array[10];
9.          char ch;
10.         scanf(x, y, z);
11.         array[5] = z;
12.         x = x / a;
13.         y = y * b;
14.         ch = array[5] - c;
15.         printf(x, y, ch);
16. }
```

错误信息
Line 14：赋值类型不匹配

（10）error5.txt

```
1.  const int a = 1;
2.  const float b = 2.0;
3.  const char c = 'c';
4.
5.  void main()
6.  {
7.          int x, y, z;
8.          int  array[10];
9.          scanf(x, y, z);
10.         array[5] = z;
11.         x = x / 1;
12.         ch = y * b;
13.         array[5] = array[5] - c;
14.         printf(x, y, array[5]);
15. }
```

错误信息
Line 12：标识符未定义

2．测试结果分析

编号	文件名	测试范围
Test1	test_ fib.txt	变量定义、函数调用、递归函数、读、写
Test2	test_ while.txt	变量定义、函数调用、循环语句、读、写
Test3	test_ switch.txt	变量定义、函数调用、情况语句、读、写
Test4	test_ max.txt	变量定义、函数调用、条件语句、读、写
Test5	test_ compute.txt	常量定义、变量定义、运算语句、读、写
Test6	error1.txt	Line 10/13/14：数组越界
Test7	error2.txt	Line 9：应以；结束
Test8	error3.txt	Line 12：常量不得赋值
Test9	error4.txt	Line 14：赋值类型不匹配
Test10	error5.txt	Line 12：标识符未定义

五．总结感想

紧张而又刺激的编译课设终于要结束了，这一学期十分充实，但又有些遗憾。

在开始之前，学长们就都纷纷提醒我，一定要尽早完成任务，赶在 DDL 之前完成会存在很大的隐患。果不其然，在词法分析和语法分析的时候，时间充裕，我都能提早完成。但等到了代码生成和解释执行时，因为其他课程的影响，真的是赶在最后一分钟才提交，导致代码中存在很多的 bug，在考试中和之后的工作中带来了很多麻烦。

不得不说，这是我目前为止写过最多的代码，三千多行的代码量，都是一点一点地码出来的，这是我之前想都不敢想的，每次 debug 的时候，都要上下来回翻阅，才能准确定位。代码量的提高意味着难度的陡增，我也为自己能完成而骄傲。

通过这一学期的课程，我感觉自己在很多方面都有了提高。

一是安排的重要性，这么庞大的工程，要是一开始就直接上手，一定会很困难。但在教学组的安排下，分为了几大部分，配合理论课程，循序渐进，感觉难度减轻了不少。相信以后的我在面对这样的大工程时，也会冷静的将任务拆分，逐步地完成。

二是坚韧的毅力，我在期间有过几次想要放弃，砸键盘的次数更是很多。尤其是每次 debug 的时候，bug 多，bug 点不明，代码量大，很容易让人产生厌烦的情绪。但好在我一直坚持了下来，课程把我磨练得更加坚定，相信这会伴随我一生。

最后是清晰的逻辑思维能力，三千多行的代码不是胡乱的编写的，是根据一套完整的架构填充的。如果没能在事先想好自己编译器结构，边写边想，不止会存在很多潜在的 bug，更危险的是存在推翻重来的危险。

非常感谢这一学期老师和助教们的帮助，祝新年快乐！