2016-2017学年第1学期

**实 验 报 告**



* 课程名称:编程语言原理与编译
* 实验项目:期末大作业
* 专业班级\_\_\_计算1501\_\_
* 学生学号\_\_\_31501337,\_\_31502176\_\_\_
* 学生姓名\_\_\_刘坤，吕奇伦\_\_\_
* 实验指导教师: 郭鸣

简介

本次实现了带类型检查的MicroC语言编译器。

**背景**

**编译原理是计算机专业的一门重要专业课，旨在介绍编译程序构造的一般原理和基本方法。内容包括语言和文法、词法分析、语法分析、语法制导翻译、中间代码生成、存储管理、代码优化和目标代码生成。如今快餐化的开发，本课程少有人研究，但是这不影响本课程在计算机领域的重要性，因此本课程大作业就是使用F#语言来实现一个micro C语言，并且做类型检查。**

**目的与目标**

实现一个较为基础的micro C语言，我们小组设计了以下功能：

1. 编译：将micro c程序编译到栈式虚拟机指令，编译保存并检查语法错误
2. 类型检查：检查基本的if，while，赋值语句，基本的运算语句等
3. 语法：主函数，带有随机空格的空主函数、带有随机空格的空主函数、二元算数表达式、一元算数表达式、返回空、返回表达式、if语句、带参数的函数、逻辑运算、if-else语句、局部声明、函数调用、赋值语句、数组。

**基本语法**

## 分号 ;

在程序中，分号是语句结束符。也就是说，每个语句必须以分号结束。它表明一个逻辑实体的结束。

例如，下面是两个不同的语句：

printf(1);  
return 0;

## 标识符

标识符是用来标识变量、函数，或任何其他用户自定义项目的名称。一个标识符以字母 A-Z 或 a-z 或下划线 \_ 开始，后跟零个或多个字母、下划线和数字（0-9）。

标识符内不允许出现标点字符，比如 @、$ 和 %。C 是**区分大小写**的编程语言。因此，在 C 中，*Manpower* 和 *manpower* 是两个不同的标识符。

空格

词法分析时，空格用来分隔两个词元，因此如下条语句：

int a;

int和a间必须有一个空格，否则编译器会认为是一个标识符。

变量

本编译器支持的变量有

|  |  |
| --- | --- |
| 类型 | 声明方式 |
| 整型 | int a; |
| 字符型 | char a; |
| 指针 | int \* a; |

**运算符**

## 算术运算符

下表显示了支持的所有算术运算符。假设变量 **A** 的值为 10，变量 **B** 的值为 20，则：

|  |  |  |
| --- | --- | --- |
| 运算符 | 描述 | 实例 |
| + | 把两个操作数相加 | A + B 将得到 30 |
| - | 从第一个操作数中减去第二个操作数 | A - B 将得到 -10 |
| \* | 把两个操作数相乘 | A \* B 将得到 200 |
| / | 分子除以分母 | B / A 将得到 2 |
| % | 取模运算符，整除后的余数 | B % A 将得到 0 |
| ++ | 自增运算符，整数值增加 1 | A++ 将得到 11 |
| -- | 自减运算符，整数值减少 1 | A-- 将得到 9 |

## 关系运算符

下表显示了支持的所有关系运算符。假设变量 **A** 的值为 10，变量 **B** 的值为 20，则：

|  |  |  |
| --- | --- | --- |
| 运算符 | 描述 | 实例 |
| == | 检查两个操作数的值是否相等，如果相等则条件为真。 | (A == B) 不为真。 |
| != | 检查两个操作数的值是否相等，如果不相等则条件为真。 | (A != B) 为真。 |
| > | 检查左操作数的值是否大于右操作数的值，如果是则条件为真。 | (A > B) 不为真。 |
| < | 检查左操作数的值是否小于右操作数的值，如果是则条件为真。 | (A < B) 为真。 |
| >= | 检查左操作数的值是否大于或等于右操作数的值，如果是则条件为真。 | (A >= B) 不为真。 |
| <= | 检查左操作数的值是否小于或等于右操作数的值，如果是则条件为真。 | (A <= B) 为真。 |

## 指针

int \* a;

上面就是指针的申明，a就是一个地址，而 \*a 用来调用指针指向的那个变量。

## 取地址符号 &

& res；

如果res是一个变量，那上面这个语句就是取res这个变量的地址，取地址符和指针经常成队出现，如下面这个实例：

void main(int n) {

int i;

i = 0;

while (i < n) { /\* Outer n \*/

int n;

fac(i, &n);

print n; /\* Inner n \*/

i = i + 1;

}

print n;

}

void fac(int n, int \*res) {

if (n == 0) /\* fac's n \*/

\*res = 1;

else {

int tmp;

fac(n-1, &tmp);

\*res = tmp \* n;

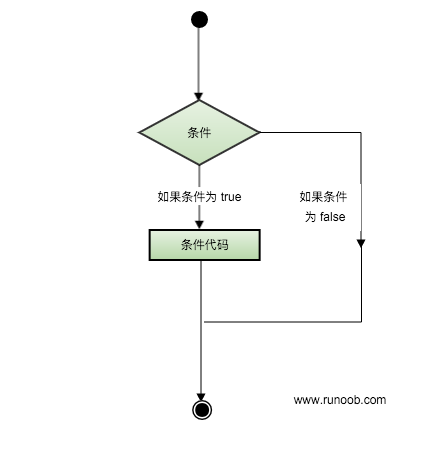
}

}

## 判断

判断结构要求程序员指定一个或多个要评估或测试的条件，以及条件为真时要执行的语句（必需的）和条件为假时要执行的语句（可选的）。

任何**非零**和**非空**的值假定为 **true**，把**零**或 **null** 假定为 **false**。

下面是大多数编程语言中典型的判断结构的一般形式：

## if 语句

if(boolean\_expression)  
{  
 /\* 如果布尔表达式为真将执行的语句 \*/  
}

如果布尔表达式为 **true**，则 if 语句内的代码块将被执行。如果布尔表达式为 **false**，则 if 语句结束后的第一组代码（闭括号后）将被执行。

## if...else 语句

if(boolean\_expression)  
{  
 /\* 如果布尔表达式为真将执行的语句 \*/  
}  
else  
{  
 /\* 如果布尔表达式为假将执行的语句 \*/  
}

如果布尔表达式为 **true**，则执行 **if** 块内的代码。如果布尔表达式为 **false**，则执行 **else** 块内的代码。

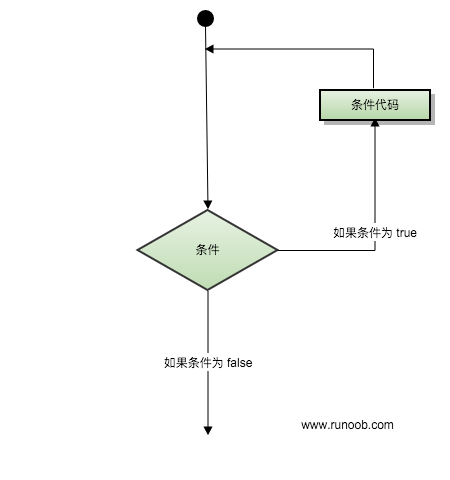
## 嵌套 if 语句

if( boolean\_expression 1)  
{  
 /\* 当布尔表达式 1 为真时执行 \*/  
 if(boolean\_expression 2)  
 {  
 /\* 当布尔表达式 2 为真时执行 \*/  
 }  
}

## 循环

有的时候，我们可能需要多次执行同一块代码。一般情况下，语句是按顺序执行的：函数中的第一个语句先执行，接着是第二个语句，依此类推。

编程语言提供了更为复杂执行路径的多种控制结构。

循环语句允许我们多次执行一个语句或语句组，下面是大多数编程语言中循环语句的流程图：

## while 循环

while(condition)  
{  
 statement(s);  
}

在这里，**statement(s)** 可以是一个单独的语句，也可以是几个语句组成的代码块。**condition** 可以是任意的表达式，当为任意非零值时都为 true。当条件为 true 时执行循环。

当条件为 false 时，程序流将继续执行紧接着循环的下一条语句。

****输出语句****

print n;

n可以是整型也可以是字符串。

## 换行符号

println;

## 函数

函数是一组一起执行一个任务的语句。每个程序都至少有一个函数，即主函数 **main()** ，所有简单的程序都可以定义其他额外的函数。

您可以把代码划分到不同的函数中。如何划分代码到不同的函数中是由您来决定的，但在逻辑上，划分通常是根据每个函数执行一个特定的任务来进行的。

函数**声明**告诉编译器函数的名称、返回类型和参数。函数**定义**提供了函数的实际主体。

## 定义函数

函数定义的一般形式如下：

return\_type function\_name( parameter list )  
{  
 body of the function  
}

函数由一个函数头和一个函数主体组成。下面列出一个函数的所有组成部分：

* **返回类型：**一个函数可以返回一个值。**return\_type** 是函数返回的值的数据类型。有些函数执行所需的操作而不返回值，在这种情况下，return\_type 是关键字 **void**。
* **函数名称：**这是函数的实际名称。函数名和参数列表一起构成了函数签名。
* **参数：**参数就像是占位符。当函数被调用时，您向参数传递一个值，这个值被称为实际参数。参数列表包括函数参数的类型、顺序、数量。参数是可选的，也就是说，函数可能不包含参数。
* **函数主体：**函数主体包含一组定义函数执行任务的语句。

## ****类型检查****

本项目类型检查是编译器之外的模块进行，操作如下：

1. 请书写一个合法的microC文件
2. 使用“fsi -r FsLexYacc.Runtime.dll Absyn.fs CPar.fs CLex.fs Parse.fs Machine.fs Comp.fs typedFun.fs ParseAndComp.fs ”语句来启动类型检查程序
3. “open ParseAndComp;; checkType "要检查的程序名";;”用来检查类型
4. “ #q;;”用来退出类型检查系统

特别注意如果程序不能通过语法分析，那么将不会进行类型检查；

如果类型检查通过不会有任何文字输出，只有当类型检查失败，才会有提示。

关键算法解释

|  |  |
| --- | --- |
| **检查句型** | **检查方法** |
| **if** | **先检查判断语句是否返回Int类型，检查分支是否有类型不匹配，返回是TypN** |
| **While** | **同if，但是while只有一个body，只需要检查body是否有类型不匹配** |
| **{}** | **这个是一个语句块，里面有stmt语句块，这里就将List遍历，分别检查stmt是否有类型不匹配。** |
| **指针检查** | **指针在抽象语法树当中是嵌套定义的，我写了一个findType方法用来将变量从环境中取出来，然后讲这个多太的变量match typeAcc with**  **| TypA (typ,\_) -> typ**  **| TypP typ -> typ**  **| \_ -> typeAcc** |
| **常见表达式** |  |
| **赋值** | **赋值的抽象语法树是以Assign为父节点，左节点是指针的引用，就是变量，有节点是表达式，这里我判断左节点的类型是否与右节点的类型相同。左右分别调用typAccess和typExpr来判定类型** |
| **单目运算符** | **单目运算父节点是Prim1，特定操作符号的参数类型是一定的，我这里使用match去匹配操作符，匹配到操作符就拿这种操作符的指定参数类型与表达式类型进行比较。匹配则为正确。** |
| **其他运算** | **几乎都是比较左右值是否类型相同** |

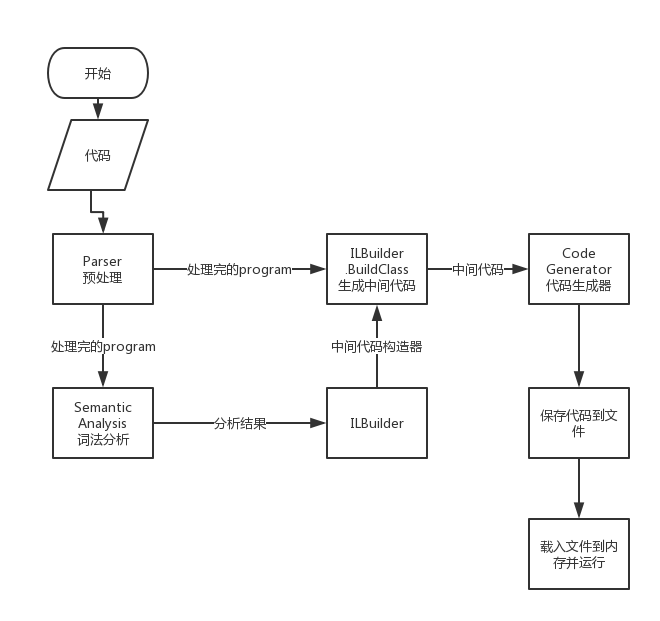
**特别说明findType这个函数**

**目的是为了找到环境当中的变量具体的类型**

**参数是变量和环境，返回值是类型**

**算法：将VarEnv分成(string \* (('a \* int) \* typ)) list和int，然后去匹配string与变量是否相等。匹配到就去把typ取出来，否则就循环调用上一步。**

**架构设计**



## ****测试****

**基于原有ex的25组代码进行测试。**

**测试的环境及启动方式：**

**CPU：Intel i5-4210H 2.90Hz 8GB**

**OS：windows 10 x64**

**启动： fslex --unicode CLex.fsl**

**fsyacc --module CPar CPar.fsy**

**//启动fsi**

**fsi -r FsLexYacc.Runtime.dll Absyn.fs CPar.fs CLex.fs Parse.fs Machine.fs Comp.fs typedFun.fs ParseAndComp.fs**

**//fsi 中输入，运行编译器**

**open ParseAndComp;;**

**checkType (fromFile "Testing Data/ex1.c");; //检查ex1.c**

**类型检查测试详细说明：**

|  |  |  |
| --- | --- | --- |
| **测试用例** | **预期结果** | **测试结果** |
| **语句测试，含指针运算**  **顺序结构程序的类型测试**  **1整型、字符型及指针赋值测试**  **void main(int n) {**  **int a;**  **int \*b;**  **a=b;**  **int c;**  **int \*d;**  **d=c;**  **char p;**  **char \*q;**  **p=q;**  **char h;**  **char \*k;**  **k=h;**  **int m;**  **int x;**  **x = (m == 0 && n == 0);**  **int y;**  **y = (m = n);**  **int z;**  **z = m = n;**  **}** | **不会报错，指针类型的实质是int** | **[Fundec**  **(None,"main",[(TypI, "n")],**  **Block**  **[Dec (TypI,"a"); Dec (TypP TypI,"b");**  **Stmt (Expr (Assign (AccVar "a",Access (AccVar "b")))); Dec (TypI,"c");**  **Dec (TypP TypI,"d");**  **Stmt (Expr (Assign (AccVar "d",Access (AccVar "c")))); Dec (TypC,"p");**  **Dec (TypP TypC,"q");**  **Stmt (Expr (Assign (AccVar "p",Access (AccVar "q")))); Dec (TypC,"h");**  **Dec (TypP TypC,"k");**  **Stmt (Expr (Assign (AccVar "k",Access (AccVar "h")))); Dec (TypI,"m");**  **Dec (TypI,"x");**  **Stmt**  **(Expr**  **(Assign**  **(AccVar "x",**  **Andalso**  **(Prim2 ("==",Access (AccVar "m"),CstI 0),**  **Prim2 ("==",Access (AccVar "n"),CstI 0))))); Dec (TypI,"y");**  **Stmt (Expr (Assign (AccVar "y",Assign (AccVar "m",Access (AccVar "n")))));**  **Dec (TypI,"z");**  **Stmt (Expr (Assign (AccVar "z",Assign (AccVar "m",Access (AccVar "n")))))])]**  **Typecheck is FINE!**  **val it : Absyn.typ = TypN**  **结果符合预期** |
| **2整型数组、整型指针数组、整型数组指针测试**  **// micro-C example 2**  **void main() {**  **int \*p;**  **int i;**  **int ia[10];**  **int\* ia2;**  **int \*ipa[10]; // array of 10 pointers to int**  **int (\*iap)[10]; // pointer to array of 10 ints**  **print i; // ~1**  **print p; // ~1**  **p = &i; // now p points to i**  **print p; // 1**  **ia2 = ia; // now ia2 points to ia[0]**  **print \*ia2; // ~1**  **\*p = 227; // now i is 227**  **print p; print i; // 1 227**  **\*&i = 12; // now i is 12**  **print i; // 12**  **p = &\*p; // no change**  **print \*p; // 12**  **p = ia; // now p points to ia[0]**  **\*ia = 14; // now ia[0] is 14**  **print ia[0]; // 14**  **\*(ia+9) = 114; // now ia[9] is 114**  **print ia[9]; // 114**  **ipa[2] = p; // now ipa[2] points to i**  **print ipa[2]; // 2**  **print (&\*ipa[2] == &\*\*(ipa+2)); // 1 (true)**  **iap = &ia; // now iap points to ia**  **print (&(\*iap)[2] == &\*((\*iap)+2)); // 1 (true)**  **}** | **没有类型错误，通过类型检查，显示Typecheck is FINE!** | **stmt:Block**  **[Dec (TypP TypI,"p"); Dec (TypI,"i"); Dec (TypA (TypI,Some 10),"ia");**  **Dec (TypP TypI,"ia2"); Dec (TypA (TypP TypI,Some 10),"ipa");**  **Dec (TypP (TypA (TypI,Some 10)),"iap");**  **Stmt (Expr (Prim1 ("printi",Access (AccVar "i"))));**  **Stmt (Expr (Prim1 ("printi",Access (AccVar "p"))));**  **Stmt (Expr (Assign (AccVar "p",Addr (AccVar "i"))));**  **Stmt (Expr (Prim1 ("printi",Access (AccVar "p"))));**  **Stmt (Expr (Assign (AccVar "ia2",Access (AccVar "ia"))));**  **Stmt (Expr (Prim1 ("printi",Access (AccDeref (Access (AccVar "ia2"))))));**  **Stmt (Expr (Assign (AccDeref (Access (AccVar "p")),CstI 227)));**  **Stmt (Expr (Prim1 ("printi",Access (AccVar "p"))));**  **Stmt (Expr (Prim1 ("printi",Access (AccVar "i"))));**  **Stmt (Expr (Assign (AccDeref (Addr (AccVar "i")),CstI 12)));**  **Stmt (Expr (Prim1 ("printi",Access (AccVar "i"))));**  **Stmt (Expr (Assign (AccVar "p",Addr (AccDeref (Access (AccVar "p"))))));**  **Stmt (Expr (Prim1 ("printi",Access (AccDeref (Access (AccVar "p"))))));**  **Stmt (Expr (Assign (AccVar "p",Access (AccVar "ia"))));**  **Stmt (Expr (Assign (AccDeref (Access (AccVar "ia")),CstI 14)));**  **Stmt (Expr (Prim1 ("printi",Access (AccIndex (AccVar "ia",CstI 0)))));**  **Stmt**  **(Expr**  **(Assign (AccDeref (Prim2 ("+",Access (AccVar "ia"),CstI 9)),CstI 114)));**  **Stmt (Expr (Prim1 ("printi",Access (AccIndex (AccVar "ia",CstI 9)))));**  **Stmt (Expr (Assign (AccIndex (AccVar "ipa",CstI 2),Access (AccVar "p"))));**  **Stmt (Expr (Prim1 ("printi",Access (AccIndex (AccVar "ipa",CstI 2)))));**  **Stmt**  **(Expr**  **(Prim1**  **("printi",**  **Prim2**  **("==",Addr (AccDeref (Access (AccIndex (AccVar "ipa",CstI 2)))),**  **Addr**  **(AccDeref**  **(Access**  **(AccDeref (Prim2 ("+",Access (AccVar "ipa"),CstI 2)))))))));**  **Stmt (Expr (Assign (AccVar "iap",Addr (AccVar "ia"))));**  **Stmt**  **(Expr**  **(Prim1**  **("printi",**  **Prim2**  **("==",Addr (AccIndex (AccDeref (Access (AccVar "iap")),CstI 2)),**  **Addr**  **(AccDeref**  **(Prim2**  **("+",Access (AccDeref (Access (AccVar "iap"))),CstI 2)))))))]**  **Typecheck is FINE!**  **val it : Absyn.typ = TypN**  **结果符合预期** |
| **分支结构类型匹配测试**  **3单目运算测试：逻辑非**  **void main(int n) {**  **char a;**  **if (!n)**  **{ }**  **if (!a)**  **{ }**  **}** | **报错，因为char类型a不能做非运算** | **WRONG INFO: Fail to operate "!" on Access (AccVar "a"), for TypC does not match {contents = TypI;}**  **System.Exception: Wrong Prim1: type of e1 and t2 dismatch**  **at FSI\_0002.TypedFun.typExpr(expr e, FSharpList`1 varEnv\_0, Int32 varEnv\_1, FSharpList`1 funEnv) in Q:\Compilation\cc\mircoC\typedFun.fs:line 150**  **……**  **Stopped due to error**  **结果符合预期** |
| **4双目运算测试：关系与算术**  **void main(int n,int m) {**  **char a;**  **if (m+n){}**  **if (m-n){}**  **if (m\*n){}**  **if (m%n){}**  **if (m/n){}**  **if (m<n){}**  **if (m>n){}**  **if (m<=n){}**  **if (m>=n){}**  **if (m==n){}**  **if (m!=n){}**  **if (m-a){}//类型间算术测试**  **if (m!=a){}//类型间关系测试**  **char b;**  **if (b+a){} //同类型算术测试**  **if (b!=a){} //同类型关系测试**  **if (b<a){}**  **if (b==a){}**  **}** | **报错，因为char和int类型间不能做所有关系算术运算，char同类型可以做关系运算，但不能做算术运算，**  **int同类型既可做算术运算也可做关系运算** | **WRONG INFO: Fail to operate "!=" on Access (AccVar "m") and Access (AccVar "a"), for TypI does not match TypC**  **System.Exception: Wrong Prim2: type of e1 and type of e2 dismatch**  **at FSI\_0002.TypedFun.typExpr(expr e, FSharpList`1 varEnv\_0, Int32 varEnv\_1, FSharpList`1 funEnv) in Q:\Compilation\cc\mircoC\typedFun.fs:line 157**  **……**  **at <StartupCode$FSI\_0034>.$FSI\_0034.main@()**  **Stopped due to error**  **但是char同类型不能做关系运算，未达到预期** |
| **5逻辑运算测试：与和或**  **void main(int x) {**  **char a;**  **if (x && a)**  **print 33;**  **if (x || a)**  **print 44;**  **}** | **报错，去掉上面的if仍然报错，因为两侧的类型不匹配** | **WRONG INFO: Operands on both sides of 'and' dismatch, for TypI does not match TypC**  **System.Exception: Diffierent type between Andalso(e1 , e2)**  **at FSI\_0002.TypedFun.typExpr(expr e, FSharpList`1 varEnv\_0, Int32 varEnv\_1, FSharpList`1 funEnv) in Q:\Compilation\cc\mircoC\typedFun.fs:line 177**  **……**  **Stopped due to error**  **WRONG INFO: Operands on both sides of 'or' dismatch, for TypI does not match TypC**  **System.Exception: Diffierent type between Orelse(e1 , e2)**  **at FSI\_0002.TypedFun.typExpr(expr e, FSharpList`1 varEnv\_0, Int32 varEnv\_1, FSharpList`1 funEnv) in Q:\Compilation\cc\mircoC\typedFun.fs:line 184**  **……**  **Stopped due to error**  **结果符合预期** |
| **循环结构类型匹配测试**  **6类型间测试**  **// micro-C example 3修改**  **void main(int n) {**  **char i;**  **i=0;**  **while (i < n) {**  **print i;**  **i=i+1;**  **}**  **}** | **报错，因为i是char类型而不是int类型** | **WRONG INFO: Fail to Assign CstI 0 to AccVar "i", for TypI does not match TypC**  **System.Exception: WRONG!**  **at FSI\_0002.TypedFun.typExpr(expr e, FSharpList`1 varEnv\_0, Int32 varEnv\_1, FSharpList`1 funEnv) in Q:\Compilation\cc\mircoC\typedFun.fs:line 136**  **……**  **at <StartupCode$FSI\_0004>.$FSI\_0004.main@()**  **Stopped due to error**  **结果符合预期** |
| **7类型内匹配测试**  **// micro-C example 13 -- optimization of andalso and orelse**  **void main(int n) {**  **int y;**  **y = 1889;**  **while (y < n) {**  **y = y + 1;**  **if (y % 4 == 0 && (y % 100 != 0 || y % 400 == 0))**  **print y;**  **}**  **}** | **没有类型错误，通过类型检查，显示Typecheck is FINE!** | **[Fundec**  **(None,"main",[(TypI, "n")],**  **Block**  **[Dec (TypI,"y"); Stmt (Expr (Assign (AccVar "y",CstI 1889)));**  **Stmt**  **(While**  **(Prim2 ("<",Access (AccVar "y"),Access (AccVar "n")),**  **Block**  **[Stmt**  **(Expr**  **(Assign (AccVar "y",Prim2 ("+",Access (AccVar "y"),CstI 1))));**  **Stmt**  **(If**  **(Andalso**  **(Prim2**  **("==",Prim2 ("%",Access (AccVar "y"),CstI 4),CstI 0),**  **Orelse**  **(Prim2**  **("!=",Prim2 ("%",Access (AccVar "y"),CstI 100),**  **CstI 0),**  **Prim2**  **("==",Prim2 ("%",Access (AccVar "y"),CstI 400),**  **CstI 0))),**  **Expr (Prim1 ("printi",Access (AccVar "y"))),Block []))]))])]**  **Typecheck is FINE!**  **val it : Absyn.typ = TypN**  **结果符合预期** |
| **函数调用测试，含参数匹配**  **8含全局数组的函数调用参数类型匹配测试**  **// micro-C example 4 -- compute and print array of factorials**  **int a[20]; /\* Must be global \*/**  **void main(int n) {**  **int i;**  **i = 0;**  **int f;**  **f = 1;**  **while (i < n) {**  **a[i] = f;**  **i = i + 1;**  **f = f \* i;**  **}**  **printarr(n, a);**  **}**  **void printarr(int len, int a[]) {**  **int i;**  **i = 0;**  **while (i < len) {**  **print a[i];**  **i=i+1;**  **}**  **}** | **没有类型错误，通过类型检查，显示Typecheck is FINE!** | **[Vardec (TypA (TypI,Some 20),"a");**  **Fundec**  **(None,"main",[(TypI, "n")],**  **Block**  **[Dec (TypI,"i"); Stmt (Expr (Assign (AccVar "i",CstI 0))); Dec (TypI,"f");**  **Stmt (Expr (Assign (AccVar "f",CstI 1)));**  **Stmt**  **(While**  **(Prim2 ("<",Access (AccVar "i"),Access (AccVar "n")),**  **Block**  **[Stmt**  **(Expr**  **(Assign**  **(AccIndex (AccVar "a",Access (AccVar "i")),**  **Access (AccVar "f"))));**  **Stmt**  **(Expr**  **(Assign (AccVar "i",Prim2 ("+",Access (AccVar "i"),CstI 1))));**  **Stmt**  **(Expr**  **(Assign**  **(AccVar "f",**  **Prim2 ("\*",Access (AccVar "f"),Access (AccVar "i")))))]));**  **Stmt**  **(Expr (Call ("printarr",[Access (AccVar "n"); Access (AccVar "a")])))]);**  **Fundec**  **(None,"printarr",[(TypI, "len"); (TypA (TypI,None), "a")],**  **Block**  **[Dec (TypI,"i"); Stmt (Expr (Assign (AccVar "i",CstI 0)));**  **Stmt**  **(While**  **(Prim2 ("<",Access (AccVar "i"),Access (AccVar "len")),**  **Block**  **[Stmt**  **(Expr**  **(Prim1**  **("printi",**  **Access (AccIndex (AccVar "a",Access (AccVar "i"))))));**  **Stmt**  **(Expr**  **(Assign (AccVar "i",Prim2 ("+",Access (AccVar "i"),CstI 1))))]))])]**  **Typecheck is FINE!**  **val it : Absyn.typ = TypN**  **结果符合预期** |
| **9在上述案例的main函数中去掉i=0，将i的类型改成char** | **报错，显示while里面类型匹配错误** | **WRONG INFO: Fail to operate "<" on Access (AccVar "i") and Access (AccVar "n"), for TypC does not match TypI**  **System.Exception: Wrong Prim2: type of e1 and type of e2 dismatch**  **at FSI\_0002.TypedFun.typExpr(expr e, FSharpList`1 varEnv\_0, Int32 varEnv\_1, FSharpList`1 funEnv) in Q:\Compilation\cc\mircoC\typedFun.fs:line 157**  **……**  **Stopped due to error**  **结果符合预期** |
| **10将上述案例改为**  **int a[20]; /\* Must be global \*/**  **void main(int n) {**  **int i;**  **i = 0;**  **int f;**  **f = 1;**  **while (i < n) {**  **a[i] = f;**  **i = i + 1;**  **f = f \* i;**  **}**  **int \* b;**  **printarr(n, b);**  **}**  **void printarr(int len, int a[]) {**  **int i;**  **i = 0;**  **while (i < len) {**  **print a[i];**  **i=i+1;**  **}**  **}**  **即将指针代替数组传入** | **没有类型错误，通过类型检查，显示Typecheck is FINE!**  **因为传入的指针与数组等价** | **[Vardec (TypA (TypI,Some 20),"a");**  **Fundec**  **(None,"main",[(TypI, "n")],**  **Block**  **[Dec (TypI,"i"); Stmt (Expr (Assign (AccVar "i",CstI 0))); Dec (TypI,"f");**  **Stmt (Expr (Assign (AccVar "f",CstI 1)));**  **Stmt**  **(While**  **(Prim2 ("<",Access (AccVar "i"),Access (AccVar "n")),**  **Block**  **[Stmt**  **(Expr**  **(Assign**  **(AccIndex (AccVar "a",Access (AccVar "i")),**  **Access (AccVar "f"))));**  **Stmt**  **(Expr**  **(Assign (AccVar "i",Prim2 ("+",Access (AccVar "i"),CstI 1))));**  **Stmt**  **(Expr**  **(Assign**  **(AccVar "f",**  **Prim2 ("\*",Access (AccVar "f"),Access (AccVar "i")))))]));**  **Dec (TypP TypI,"b");**  **Stmt**  **(Expr (Call ("printarr",[Access (AccVar "n"); Access (AccVar "b")])))]);**  **Fundec**  **(None,"printarr",[(TypI, "len"); (TypA (TypI,None), "a")],**  **Block**  **[Dec (TypI,"i"); Stmt (Expr (Assign (AccVar "i",CstI 0)));**  **Stmt**  **(While**  **(Prim2 ("<",Access (AccVar "i"),Access (AccVar "len")),**  **Block**  **[Stmt**  **(Expr**  **(Prim1**  **("printi",**  **Access (AccIndex (AccVar "a",Access (AccVar "i"))))));**  **Stmt**  **(Expr**  **(Assign (AccVar "i",Prim2 ("+",Access (AccVar "i"),CstI 1))))]))])]**  **Typecheck is FINE!**  **val it : Absyn.typ = TypN**  **结果符合预期** |
| **11含指针函数调用类型检查**  **// micro-C example 5 -- return a result via a pointer argument; nested blocks**  **void main(int n) {**  **int r;**  **r = n;**  **{**  **int r;**  **square(n, &r);**  **print r;**  **}**  **print r;**  **}**  **void square(int i, char \*rp) {**  **\*rp = i \* i;**  **}** | **没有类型错误，通过类型检查，显示Typecheck is FINE!** | **[Fundec**  **(None,"main",[(TypI, "n")],**  **Block**  **[Dec (TypI,"r"); Stmt (Expr (Assign (AccVar "r",Access (AccVar "n"))));**  **Stmt**  **(Block**  **[Dec (TypI,"r");**  **Stmt**  **(Expr (Call ("square",[Access (AccVar "n"); Addr (AccVar "r")])));**  **Stmt (Expr (Prim1 ("printi",Access (AccVar "r"))))]);**  **Stmt (Expr (Prim1 ("printi",Access (AccVar "r"))))]);**  **Fundec**  **(None,"square",[(TypI, "i"); (TypP TypI, "rp")],**  **Block**  **[Stmt**  **(Expr**  **(Assign**  **(AccDeref (Access (AccVar "rp")),**  **Prim2 ("\*",Access (AccVar "i"),Access (AccVar "i")))))])]**  **Typecheck is FINE!**  **val it : Absyn.typ = TypN**  **结果符合预期** |
| **12分支语句函数调用单目运算测试**  **//ex17.c改**  **void main() {**  **int k;**  **k = 0;**  **while (true) {**  **k = k+1;**  **if (!seq(k))**  **print k;**  **if (!sec(k))**  **print k;**  **}**  **}**  **int seq(int i) {**  **int count;**  **count = 0;**  **while (i != 1) {**  **count = count + 1;**  **if (i % 2 == 0)**  **i = i / 2;**  **else**  **i = i \* 3 + 1;**  **}**  **return count;**  **}**  **char sec(int i){**  **char a;**  **return a;**  **}** | **报错，sec(k)返回char，不能用！运算** | **[Fundec**  **(None,"main",[],**  **Block**  **[Dec (TypI,"k"); Stmt (Expr (Assign (AccVar "k",CstI 0)));**  **Stmt**  **(While**  **(CstI 1,**  **Block**  **[Stmt**  **(Expr**  **(Assign (AccVar "k",Prim2 ("+",Access (AccVar "k"),CstI 1))));**  **Stmt**  **(If**  **(Prim1 ("!",Call ("seq",[Access (AccVar "k")])),**  **Expr (Prim1 ("printi",Access (AccVar "k"))),Block []));**  **Stmt**  **(If**  **(Prim1 ("!",Call ("sec",[Access (AccVar "k")])),**  **Expr (Prim1 ("printi",Access (AccVar "k"))),Block []))]))]);**  **Fundec**  **(Some TypI,"seq",[(TypI, "i")],**  **Block**  **[Dec (TypI,"count"); Stmt (Expr (Assign (AccVar "count",CstI 0)));**  **Stmt**  **(While**  **(Prim2 ("!=",Access (AccVar "i"),CstI 1),**  **Block**  **[Stmt**  **(Expr**  **(Assign**  **(AccVar "count",**  **Prim2 ("+",Access (AccVar "count"),CstI 1))));**  **Stmt**  **(If**  **(Prim2 ("==",Prim2 ("%",Access (AccVar "i"),CstI 2),CstI 0),**  **Expr**  **(Assign**  **(AccVar "i",Prim2 ("/",Access (AccVar "i"),CstI 2))),**  **Expr**  **(Assign**  **(AccVar "i",**  **Prim2**  **("+",Prim2 ("\*",Access (AccVar "i"),CstI 3),CstI 1)))))]));**  **Stmt (Return (Some (Access (AccVar "count"))))]);**  **Fundec**  **(Some TypC,"sec",[(TypI, "i")],**  **Block [Dec (TypC,"a"); Stmt (Return (Some (Access (AccVar "a"))))])]**  **Typecheck is FINE!**  **val it : Absyn.typ = TypN**  **不符合预期** |
| **13分支语句函数调用双目运算测试**  **void main() {**  **int k;**  **k = 0;**  **while (true) {**  **k = k+1;**  **if (seq(k) > 240)**  **print k;**  **if (sec(k) > 240)**  **print k;**  **}**  **}**  **int seq(int i) {**  **int count;**  **count = 0;**  **while (i != 1) {**  **count = count + 1;**  **if (i % 2 == 0)**  **i = i / 2;**  **else**  **i = i \* 3 + 1;**  **}**  **return count;**  **}**  **char sec(int i){**  **char a;**  **return a;**  **}** | **报错，sec(k)返回char，不能用>运算** | **[Fundec**  **(None,"main",[],**  **Block**  **[Dec (TypI,"k"); Stmt (Expr (Assign (AccVar "k",CstI 0)));**  **Stmt**  **(While**  **(CstI 1,**  **Block**  **[Stmt**  **(Expr**  **(Assign (AccVar "k",Prim2 ("+",Access (AccVar "k"),CstI 1))));**  **Stmt**  **(If**  **(Prim2 (">",Call ("seq",[Access (AccVar "k")]),CstI 240),**  **Expr (Prim1 ("printi",Access (AccVar "k"))),Block []));**  **Stmt**  **(If**  **(Prim2 (">",Call ("sec",[Access (AccVar "k")]),CstI 240),**  **Expr (Prim1 ("printi",Access (AccVar "k"))),Block []))]))]);**  **Fundec**  **(Some TypI,"seq",[(TypI, "i")],**  **Block**  **[Dec (TypI,"count"); Stmt (Expr (Assign (AccVar "count",CstI 0)));**  **Stmt**  **(While**  **(Prim2 ("!=",Access (AccVar "i"),CstI 1),**  **Block**  **[Stmt**  **(Expr**  **(Assign**  **(AccVar "count",**  **Prim2 ("+",Access (AccVar "count"),CstI 1))));**  **Stmt**  **(If**  **(Prim2 ("==",Prim2 ("%",Access (AccVar "i"),CstI 2),CstI 0),**  **Expr**  **(Assign**  **(AccVar "i",Prim2 ("/",Access (AccVar "i"),CstI 2))),**  **Expr**  **(Assign**  **(AccVar "i",**  **Prim2**  **("+",Prim2 ("\*",Access (AccVar "i"),CstI 3),CstI 1)))))]));**  **Stmt (Return (Some (Access (AccVar "count"))))]);**  **Fundec**  **(Some TypC,"sec",[(TypI, "i")],**  **Block [Dec (TypC,"a"); Stmt (Return (Some (Access (AccVar "a"))))])]**  **Typecheck is FINE!**  **val it : Absyn.typ = TypN**  **不符合预期** |
| **14函数调用后指针指向变化的类型匹配测试**  **// micro-C example 9 -- return a result via a pointer argument修改**  **int t;**  **void main(int i) {**  **char r;**  **fac(i, &r);**  **print r;**  **}**  **void fac(int n, int \*res) {**  **// print &n; // Show n's address**  **if (n == 0)**  **\*res = 1;**  **else {**  **int tmp;**  **fac(n - 1, &tmp);**  **\*res = tmp \* n;**  **}**  **}** | **报错，因为r经过函数调用后，r的取值的定义为int，print时类型int与而主函数中r定义的char类型不匹配** | **WRONG INFO: Fail to operate "printi" on Access (AccVar "r"), for TypC does not match {contents = TypI;}**  **System.Exception: Wrong Prim1: type of e1 and t2 dismatch**  **at FSI\_0002.TypedFun.typExpr(expr e, FSharpList`1 varEnv\_0, Int32 varEnv\_1, FSharpList`1 funEnv) in Q:\Compilation\cc\mircoC\typedFun.fs:line 150**  **……**  **at <StartupCode$FSI\_0018>.$FSI\_0018.main@()**  **Stopped due to error**  **结果符合预期** |
| **15复杂的return语句与其返回类型的匹配**  **// micro-C example 22 -- leapyear, optimization of andalso and orelse**  **void main(int n) {**  **int y;**  **y = 1889;**  **while (y < n) {**  **y = y + 1;**  **if (leapyear(y))**  **print y;**  **}**  **}**  **int leapyear(int y) {**  **return y % 4 == 0 && (y % 100 != 0 || y % 400 == 0);**  **}**  **char leapyear1(int y) {**  **return y % 4 == 0 && (y % 100 != 0 || y % 400 == 0);**  **}** | **没有类型错误，通过类型检查，显示Typecheck is FINE!** | **[Fundec**  **(None,"main",[(TypI, "n")],**  **Block**  **[Dec (TypI,"y"); Stmt (Expr (Assign (AccVar "y",CstI 1889)));**  **Stmt**  **(While**  **(Prim2 ("<",Access (AccVar "y"),Access (AccVar "n")),**  **Block**  **[Stmt**  **(Expr**  **(Assign (AccVar "y",Prim2 ("+",Access (AccVar "y"),CstI 1))));**  **Stmt**  **(If**  **(Call ("leapyear",[Access (AccVar "y")]),**  **Expr (Prim1 ("printi",Access (AccVar "y"))),Block []))]))]);**  **Fundec**  **(Some TypI,"leapyear",[(TypI, "y")],**  **Block**  **[Stmt**  **(Return**  **(Some**  **(Andalso**  **(Prim2 ("==",Prim2 ("%",Access (AccVar "y"),CstI 4),CstI 0),**  **Orelse**  **(Prim2**  **("!=",Prim2 ("%",Access (AccVar "y"),CstI 100),CstI 0),**  **Prim2**  **("==",Prim2 ("%",Access (AccVar "y"),CstI 400),CstI 0))))))])]**  **Typecheck is FINE!**  **val it : Absyn.typ = TypN** |
| **作用域测试**  **16不同作用域同名变量类型匹配测试**  **// micro-C example 6 -- return a result via a pointer argument; nested blocks**  **void main(int n) {**  **int i;**  **i = 0;**  **while (i < n) { /\* Outer n \*/**  **int n;**  **fac(i, &n);**  **print n; /\* Inner n \*/**  **i = i + 1;**  **}**  **print n;**  **}**  **void fac(int n, int \*res) {**  **if (n == 0) /\* fac's n \*/**  **\*res = 1;**  **else {**  **int tmp;**  **fac(n-1, &tmp);**  **\*res = tmp \* n;**  **}**  **}** | **没有类型错误，通过类型检查，显示Typecheck is FINE!** | **[Fundec**  **(None,"main",[(TypI, "n")],**  **Block**  **[Dec (TypI,"i"); Stmt (Expr (Assign (AccVar "i",CstI 0)));**  **Stmt**  **(While**  **(Prim2 ("<",Access (AccVar "i"),Access (AccVar "n")),**  **Block**  **[Dec (TypI,"n");**  **Stmt**  **(Expr (Call ("fac",[Access (AccVar "i"); Addr (AccVar "n")])));**  **Stmt (Expr (Prim1 ("printi",Access (AccVar "n"))));**  **Stmt**  **(Expr**  **(Assign (AccVar "i",Prim2 ("+",Access (AccVar "i"),CstI 1))))]));**  **Stmt (Expr (Prim1 ("printi",Access (AccVar "n"))))]);**  **Fundec**  **(None,"fac",[(TypI, "n"); (TypP TypI, "res")],**  **Block**  **[Stmt**  **(If**  **(Prim2 ("==",Access (AccVar "n"),CstI 0),**  **Expr (Assign (AccDeref (Access (AccVar "res")),CstI 1)),**  **Block**  **[Dec (TypI,"tmp");**  **Stmt**  **(Expr**  **(Call**  **("fac",**  **[Prim2 ("-",Access (AccVar "n"),CstI 1);**  **Addr (AccVar "tmp")])));**  **Stmt**  **(Expr**  **(Assign**  **(AccDeref (Access (AccVar "res")),**  **Prim2 ("\*",Access (AccVar "tmp"),Access (AccVar "n")))))]))])]**  **Typecheck is FINE!**  **val it : Absyn.typ = TypN**  **结果符合预期** |

**测试中的bug**

**1. 当代码中while语句中出现单引号字符时，词法器无法识别。**

|  |  |
| --- | --- |
| **void main(int n) {**  **char i;**  **//i = 0;**  **int f;**  **f = 1;**  **while (i < 'n') {**  **a[i] = f;**  **i = i + 1;**  **f = f \* i;**  **}**  **}** | **System.Exception: Lexer error: illegal symbol in file Testing Data/ex4.c near line 10, column 14**  **at FSI\_0002.Parse.fromFile@31-1.Invoke(String message) in Q:\Compilation\cc\mircoC\Parse.fs:line 31**  **at FSI\_0002.Parse.fromFile(String filename) in Q:\Compilation\cc\mircoC\Parse.fs:line 31**  **at FSI\_0002.ParseAndComp.fromFile@9-6.Invoke(String filename) in Q:\Compilation\cc\mircoC\ParseAndComp.fs:line 9**  **at <StartupCode$FSI\_0009>.$FSI\_0009.main@()**  **Stopped due to error** |

**2. 嵌套代码块不同类型的同名变量print操作结果未正确显示，而显示代码块中的变量内容。**

|  |  |
| --- | --- |
| **void main(int n) {**  **int r;**  **r = n;**  **{**  **char r;**  **square(n, &r);**  **print r;**  **}**  **print r;**  **}**  **void square(int i, int \*rp) {**  **\*rp = i \* i;**  **}** | **WRONG INFO: Fail to operate "printi" on Access (AccVar "r"), for TypC does not match {contents = TypI;}**  **System.Exception: Wrong Prim1: type of e1 and t2 dismatch**  **at FSI\_0002.TypedFun.typExpr(expr e, FSharpList`1 varEnv\_0, Int32 varEnv\_1, FSharpList`1 funEnv) in Q:\Compilation\cc\mircoC\typedFun.fs:line 150**  **at FSI\_0002.TypedFun.typStmtOrDec(stmtordec stmtOrDec, FSharpList`1 varEnv\_0, Int32 varEnv\_1, FSharpList`1 funEnv) in Q:\Compilation\cc\mircoC\typedFun.fs:line 212**  **at FSI\_0002.TypedFun.loop@240-1(FSharpList`1 funEnv, FSharpList`1 stmts, Tuple`2 varEnv) in Q:\Compilation\cc\mircoC\typedFun.fs:line 0**  **at FSI\_0002.TypedFun.loop@240-1(FSharpList`1 funEnv, FSharpList`1 stmts, Tuple`2 varEnv) in Q:\Compilation\cc\mircoC\typedFun.fs:line 249**  **at FSI\_0002.TypedFun.loop@240-1(FSharpList`1 funEnv, FSharpList`1 stmts, Tuple`2 varEnv) in Q:\Compilation\cc\mircoC\typedFun.fs:line 249**  **at FSI\_0002.TypedFun.typStmt(stmt e, FSharpList`1 varEnv\_0, Int32 varEnv\_1, FSharpList`1 funEnv) in Q:\Compilation\cc\mircoC\typedFun.fs:line 251**  **at FSI\_0002.TypedFun.typStmtOrDec(stmtordec stmtOrDec, FSharpList`1 varEnv\_0, Int32 varEnv\_1, FSharpList`1 funEnv) in Q:\Compilation\cc\mircoC\typedFun.fs:line 212**  **at FSI\_0002.TypedFun.loop@240-1(FSharpList`1 funEnv, FSharpList`1 stmts, Tuple`2 varEnv) in Q:\Compilation\cc\mircoC\typedFun.fs:line 0**  **at FSI\_0002.TypedFun.loop@240-1(FSharpList`1 funEnv, FSharpList`1 stmts, Tuple`2 varEnv) in Q:\Compilation\cc\mircoC\typedFun.fs:line 249**  **at FSI\_0002.TypedFun.loop@240-1(FSharpList`1 funEnv, FSharpList`1 stmts, Tuple`2 varEnv) in Q:\Compilation\cc\mircoC\typedFun.fs:line 249**  **at FSI\_0002.TypedFun.typStmt(stmt e, FSharpList`1 varEnv\_0, Int32 varEnv\_1, FSharpList`1 funEnv) in Q:\Compilation\cc\mircoC\typedFun.fs:line 251**  **at FSI\_0002.TypedFun.cherkfun@262[a,b,c](FSharpList`1 globalVarEnv, FSharpList`1 funEnv, a tupledArg0, String tupledArg1, b tupledArg2, stmt tupledArg3) in Q:\Compilation\cc\mircoC\typedFun.fs:line 266**  **at FSI\_0002.TypedFun.typTopdec@269.Invoke(topdec \_arg2) in Q:\Compilation\cc\mircoC\typedFun.fs:line 271**  **at Microsoft.FSharp.Primitives.Basics.List.choose[T,TResult](FSharpFunc`2 f, FSharpList`1 xs)**  **at FSI\_0002.TypedFun.typTopdec(program \_arg1) in Q:\Compilation\cc\mircoC\typedFun.fs:line 260**  **at FSI\_0002.ParseAndComp.checkType@13.Invoke(program arg00@) in Q:\Compilation\cc\mircoC\ParseAndComp.fs:line 13**  **at <StartupCode$FSI\_0011>.$FSI\_0011.main@()**  **Stopped due to error** |

**3.指针不应被直接赋值**

|  |  |
| --- | --- |
| **void main() {**  **int \*i;**  **i = 20000000;**  **while (i) {**  **i = i - 1;**  **}**  **}** | **[Fundec**  **(None,"main",[],**  **Block**  **[Dec (TypP TypI,"i"); Stmt (Expr (Assign (AccVar "i",CstI 20000000)));**  **Stmt**  **(While**  **(Access (AccVar "i"),**  **Block**  **[Stmt**  **(Expr**  **(Assign (AccVar "i",Prim2 ("-",Access (AccVar "i"),CstI 1))))]))])]**  **Typecheck is FINE!**  **val it : Absyn.typ = TypN** |

**4. 嵌套代码块定义与函数参数同名但不同类型的变量不应报错**

|  |  |
| --- | --- |
| **// micro-C example 10 -- return a result from function; nested blocks**  **void main(int n) {**  **int i;**  **i = 0;**  **while (i <= n) {**  **print fac(i);**  **i = i + 1;**  **}**  **{**  **char n;**  **print n;**  **}**  **}**  **int fac(int n) {**  **if (n == 0) /\* fac's n \*/**  **return 1;**  **else**  **return n \* fac(n-1);**  **}** | **WRONG INFO: Fail to operate "printi" on Access (AccVar "n"), for TypC does not match {contents = TypI;}**  **System.Exception: Wrong Prim1: type of e1 and t2 dismatch**  **at FSI\_0002.TypedFun.typExpr(expr e, FSharpList`1 varEnv\_0, Int32 varEnv\_1, FSharpList`1 funEnv) in Q:\Compilation\cc\mircoC\typedFun.fs:line 150**  **at FSI\_0002.TypedFun.typStmtOrDec(stmtordec stmtOrDec, FSharpList`1 varEnv\_0, Int32 varEnv\_1, FSharpList`1 funEnv) in Q:\Compilation\cc\mircoC\typedFun.fs:line 212**  **at FSI\_0002.TypedFun.loop@240-1(FSharpList`1 funEnv, FSharpList`1 stmts, Tuple`2 varEnv) in Q:\Compilation\cc\mircoC\typedFun.fs:line 0**  **at FSI\_0002.TypedFun.loop@240-1(FSharpList`1 funEnv, FSharpList`1 stmts, Tuple`2 varEnv) in Q:\Compilation\cc\mircoC\typedFun.fs:line 249**  **at FSI\_0002.TypedFun.typStmt(stmt e, FSharpList`1 varEnv\_0, Int32 varEnv\_1, FSharpList`1 funEnv) in Q:\Compilation\cc\mircoC\typedFun.fs:line 251**  **at FSI\_0002.TypedFun.typStmtOrDec(stmtordec stmtOrDec, FSharpList`1 varEnv\_0, Int32 varEnv\_1, FSharpList`1 funEnv) in Q:\Compilation\cc\mircoC\typedFun.fs:line 212**  **at FSI\_0002.TypedFun.loop@240-1(FSharpList`1 funEnv, FSharpList`1 stmts, Tuple`2 varEnv) in Q:\Compilation\cc\mircoC\typedFun.fs:line 0**  **at FSI\_0002.TypedFun.loop@240-1(FSharpList`1 funEnv, FSharpList`1 stmts, Tuple`2 varEnv) in Q:\Compilation\cc\mircoC\typedFun.fs:line 249**  **at FSI\_0002.TypedFun.loop@240-1(FSharpList`1 funEnv, FSharpList`1 stmts, Tuple`2 varEnv) in Q:\Compilation\cc\mircoC\typedFun.fs:line 249**  **at FSI\_0002.TypedFun.loop@240-1(FSharpList`1 funEnv, FSharpList`1 stmts, Tuple`2 varEnv) in Q:\Compilation\cc\mircoC\typedFun.fs:line 249**  **at FSI\_0002.TypedFun.typStmt(stmt e, FSharpList`1 varEnv\_0, Int32 varEnv\_1, FSharpList`1 funEnv) in Q:\Compilation\cc\mircoC\typedFun.fs:line 251**  **at FSI\_0002.TypedFun.cherkfun@262[a,b,c](FSharpList`1 globalVarEnv, FSharpList`1 funEnv, a tupledArg0, String tupledArg1, b tupledArg2, stmt tupledArg3) in Q:\Compilation\cc\mircoC\typedFun.fs:line 266**  **at FSI\_0002.TypedFun.typTopdec@269.Invoke(topdec \_arg2) in Q:\Compilation\cc\mircoC\typedFun.fs:line 271**  **at Microsoft.FSharp.Primitives.Basics.List.choose[T,TResult](FSharpFunc`2 f, FSharpList`1 xs)**  **at FSI\_0002.TypedFun.typTopdec(program \_arg1) in Q:\Compilation\cc\mircoC\typedFun.fs:line 260**  **at FSI\_0002.ParseAndComp.checkType@13.Invoke(program arg00@) in Q:\Compilation\cc\mircoC\ParseAndComp.fs:line 13**  **at <StartupCode$FSI\_0020>.$FSI\_0020.main@()**  **Stopped due to error** |

**5.指针应该不能进行求余运算**

|  |  |
| --- | --- |
| **void main(int n) {**  **int y;**  **y = 1889;**  **while (y < n) {**  **int \* y;**  **y = y + 1;**  **if (y % 4 == 0 && (y % 100 != 0 || y % 400 == 0))**  **print y;**  **}**  **}** | **[Fundec**  **(None,"main",[(TypI, "n")],**  **Block**  **[Dec (TypI,"y"); Stmt (Expr (Assign (AccVar "y",CstI 1889)));**  **Stmt**  **(While**  **(Prim2 ("<",Access (AccVar "y"),Access (AccVar "n")),**  **Block**  **[Dec (TypP TypI,"y");**  **Stmt**  **(Expr**  **(Assign (AccVar "y",Prim2 ("+",Access (AccVar "y"),CstI 1))));**  **Stmt**  **(If**  **(Andalso**  **(Prim2**  **("==",Prim2 ("%",Access (AccVar "y"),CstI 4),CstI 0),**  **Orelse**  **(Prim2**  **("!=",Prim2 ("%",Access (AccVar "y"),CstI 100),**  **CstI 0),**  **Prim2**  **("==",Prim2 ("%",Access (AccVar "y"),CstI 400),**  **CstI 0))),**  **Expr (Prim1 ("printi",Access (AccVar "y"))),Block []))]))])]**  **Typecheck is FINE!**  **val it : Absyn.typ = TypN** |

**6.指针应不能与自身相加**

|  |  |
| --- | --- |
| **void main() {**  **int a;**  **a = 1;**  **int b;**  **b=a+&a;**  **int \*c;**  **int d;**  **d=c+&c+\*c;**  **}** | **[Fundec**  **(None,"main",[],**  **Block**  **[Dec (TypI,"a"); Stmt (Expr (Assign (AccVar "a",CstI 1))); Dec (TypI,"b");**  **Stmt**  **(Expr**  **(Assign**  **(AccVar "b",Prim2 ("+",Access (AccVar "a"),Addr (AccVar "a")))));**  **Dec (TypP TypI,"c"); Dec (TypI,"d");**  **Stmt**  **(Expr**  **(Assign**  **(AccVar "d",**  **Prim2**  **("+",Prim2 ("+",Access (AccVar "c"),Addr (AccVar "c")),**  **Access (AccDeref (Access (AccVar "c")))))))])]**  **Typecheck is FINE!**  **val it : Absyn.typ = TypN** |

**7.return返回值与函数返回类型未匹配，函数未找到却不报错**

|  |  |
| --- | --- |
| **void main(int n) {**  **int y;**  **y = 1889;**  **while (y < n) {**  **y = y + 1;**  **if (leapyear(y))**  **print y;**  **if(leapyear1(y))**  **print y;**  **}**  **}**  **int leapyear(int y) {**  **return y % 4 == 0 && (y % 100 != 0 || y % 400 == 0);**  **}**  **char leapyear1(int y) {**  **return y % 4 == 0 && (y % 100 != 0 || y % 400 == 0);**  **}** | **[Fundec**  **(None,"main",[(TypI, "n")],**  **Block**  **[Dec (TypI,"y"); Stmt (Expr (Assign (AccVar "y",CstI 1889)));**  **Stmt**  **(While**  **(Prim2 ("<",Access (AccVar "y"),Access (AccVar "n")),**  **Block**  **[Stmt**  **(Expr**  **(Assign (AccVar "y",Prim2 ("+",Access (AccVar "y"),CstI 1))));**  **Stmt**  **(If**  **(Call ("leapyear",[Access (AccVar "y")]),**  **Expr (Prim1 ("printi",Access (AccVar "y"))),Block []));**  **Stmt**  **(If**  **(Call ("leapyear1",[Access (AccVar "y")]),**  **Expr (Prim1 ("printi",Access (AccVar "y"))),Block []))]))]);**  **Fundec**  **(Some TypI,"leapyear",[(TypI, "y")],**  **Block**  **[Stmt**  **(Return**  **(Some**  **(Andalso**  **(Prim2 ("==",Prim2 ("%",Access (AccVar "y"),CstI 4),CstI 0),**  **Orelse**  **(Prim2**  **("!=",Prim2 ("%",Access (AccVar "y"),CstI 100),CstI 0),**  **Prim2**  **("==",Prim2 ("%",Access (AccVar "y"),CstI 400),CstI 0))))))]);**  **Fundec**  **(Some TypC,"leapyear1",[(TypI, "y")],**  **Block**  **[Stmt**  **(Return**  **(Some**  **(Andalso**  **(Prim2 ("==",Prim2 ("%",Access (AccVar "y"),CstI 4),CstI 0),**  **Orelse**  **(Prim2**  **("!=",Prim2 ("%",Access (AccVar "y"),CstI 100),CstI 0),**  **Prim2**  **("==",Prim2 ("%",Access (AccVar "y"),CstI 400),CstI 0))))))])]**  **Typecheck is FINE!**  **val it : Absyn.typ = TypN** |

## ****心得体会****

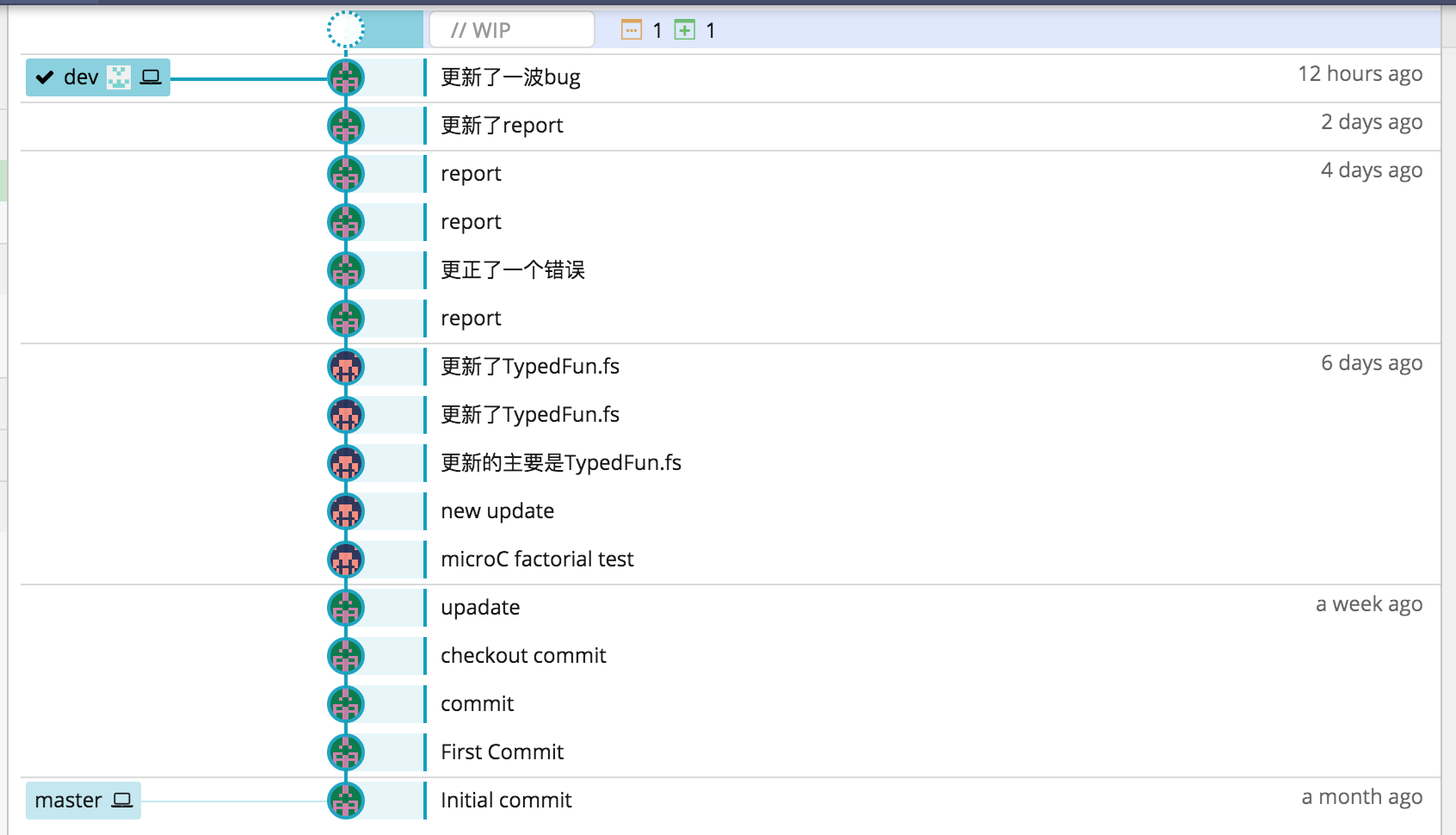
**刘坤：**

**上完编译原理之后其实是比较懵逼的，这是一门很低层的学科，他做了将高级语言向机器语言转换的工作，使得程序员不需要写汇编语言或者更低级的01序列。这门课非常难，以至于长时间处于不能理解的状态。但是通过对每一个实验的学习，我认识了这门课程的冰山一角。在学期末，大概一个月的时间完成了类型检查模块的编写，这个模块当然不是很完美，特别是一开始永远检查不到错误，真让人沮丧。然后，说实话fsharp这门语言真的非常难学，没有c语言逻辑清晰，也没有java那么好理解，希望是真心希望老师以后开着门课不要选择fsharp，入门实在是太难了。还有，这门课最好还是考试吧，由于比较偏理论，其实自己开发一个编译器成功与否都是未知的。**

**吕奇伦：**

**当时选择编译原理是为了能让自己建立一套从编码、编译、汇编到链接的完整的计算机知识体系。前面几节课的语言原理和词法语法部分理论性较强，我对它逻辑语言描述的方式感到挺有兴趣。但是在语义分析之后，直到大作业，我都没能将所有的知识点一一搞明白，这次的大作业我们在microC既定代码的基础之上增加了类型检查，在编写的过程中，虽然遇到了十分陌生的语言，但我们小组却配合得很默契，版本管理得当，队友为项目付出巨大心血，衷心地感谢他。最后我们无论将编译器完成到哪种状态都将成为一次宝贵的开发经验，感谢老师也能在每次的作业中列举很多文献资料，能够在增大阅读量的同时提升英文水平，对于我出国深造是非常有利的。**

**Git截图**

****

**计算评价表**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 一般 | 中 | 良 | 优 |
| 编译并在内存中运行 |  |  | o |  |
| 编译保存并执行 |  |  |  | o |
| 编译: 检测语法错误 |  |  | o |  |
| 基本语法 |  |  |  | o |
| 二元表达式 |  |  | o |  |
| 二元算数表达式 |  |  |  | o |
| 语法: if-else语句 |  |  |  | O |
| 语法：while语句 |  |  |  | o |
| 语法: 数组 |  |  | o |  |
| 类型检查 |  | o |  |  |

**分工表**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 姓名 | 班级 | 学号 | 任务 | 比重 |
| 刘坤 | 计算1501 | 31501337 | 大部分编码 | 0.95 |
| 吕奇伦 | 计算1501 | 31502176 | 测试及文档 | 0.95 |