编译原理课程设计

本项目为编译原理课程设计的相关代码与文档。

语法

- <程序> → <定义语句列表>
- <定义语句列表> → <定义语句><定义语句列表> | <定义语句>
- <定义语句> → <函数定义> | <变量定义语句> | <常量定义语句>
- <函数定义> → fun <标识符>(<参数列表>) : <类型> <代码块>
- <参数列表> → ϵ | <标识符> : <类型>, <参数列表>
- <代码块> → {<语句列表>}
- <语句列表> → ϵ | <语句> <语句列表> | <代码块><语句列表>
- <语句> → <变量定义语句> | <常量定义语句> | <表达式语句> | <if 语句> | <while 语句> | <返回语句>
- <变量定义语句> → var <初始化列表> | <标识符列表> : <类型>;
- <常量定义语句> → val <初始化列表>;
- <初始化列表> → <标识符> = <表达式> | <标识符> = <表达式>, <初始化列表>
- <标识符列表> → <标识符> | <标识符>, <标识符列表>
- <表达式语句> → <表达式>;
- <表达式> → <赋值表达式>;
- <赋值表达式> → <变量标识符> = <赋值表达式> | <逻辑表达式>
- <逻辑表达式> → <逻辑或表达式>
- <逻辑或表达式> → <逻辑与表达式> | <逻辑或表达式> || <逻辑与表达式>
- <逻辑与表达式> → <逻辑非表达式> | <逻辑与表达式> && <逻辑非表达达式>
- <逻辑非表达式> → !<关系表达式> | <关系表达式>
- <关系表达式> → <算术表达式> <关系运算符> <算术表达式> | <算术表达式>
- <算术表达式> → <项> | <算术表达式> <加减运算符> <项>
- <项> → <原子表达式> | <项> <乘除运算符> <原子表达式>
- <原子表达式> → <变量标识符> | <常量标识符> | <函数调用表达式> | <类型转换表达式> | <常数>
- <类型转换表达式> → <类型>(<表达式>)
- <函数调用表达式> → <函数标识符>(<实参列表>)
- <if 语句> → if (<表达式>) <代码块> | if (<表达式>) <代码块> else <代码块>
- <while 语句> → while (<表达式>) <代码块>
- <返回语句> → return <表达式>;
- <实参列表> → ϵ | <表达式>, <实参列表>
- <类型> → Int | Float | Char | Bool
- <乘除运算符> → $*$ | $/$
- <加减运算符> → $+$ | $-$
- <关系运算符> → $==$ | $!=$ | $<$ | $<=$ | $>$ | $>=$

改造后语法

- <程序> → <定义语句列表> ①
- <定义语句列表> → <定义语句><定义语句列表'> ①
- <定义语句列表'> → <定义语句><定义语句列表'> ① | ϵ
- <定义语句> → <函数定义> ① | <变量定义语句> ② | <常量定义语句> ③
- <函数定义> → fun <标识符>(<参数列表>) : <类型> <代码块> ①

- <参数列表> → ϵ ① | <标识符> : <类型>, <参数列表> ②
- <代码块> → {<语句列表>} ①
- <语句列表> → ϵ ① | <语句> <语句列表> ② | <代码块><语句列表> ③
- <语句> → <变量定义语句> ① | <常量定义语句> ② | <表达式语句> ③ | <if 语句> ④ | <while 语句> ⑤ | <返回语句> ⑥
- <变量定义语句> → var <标识符><变量定义后缀> ①
- <变量定义后缀> → <初始化列表后缀> ① | <标识符列表后缀> ②
- <初始化列表后缀> → = <表达式><初始化列表后缀'> ①
- <初始化列表后缀'> → ;① | ,<标识符> = <表达式><初始化列表后缀'> ②
- <标识符列表后缀> → : <类型>; ① | ,<标识符><标识符列表后缀> ②
- <常量定义语句> → val <初始化列表>; ①
- <初始化列表> → <标识符> = <表达式><初始化列表'> ①
- <初始化列表'> → ϵ | <标识符> = <表达式>, <初始化列表'> ①
- <表达式语句> → <表达式>; ①
- <表达式> → <赋值表达式>
- <赋值表达式> → <变量标识符> = <赋值表达式> | <逻辑表达式>
- <逻辑表达式> → <逻辑或表达式>
- <逻辑或表达式> → <逻辑与表达式> | <逻辑或表达式> || <逻辑与表达式>
- <逻辑与表达式> → <逻辑非表达式> | <逻辑与表达式> && <逻辑非表达式>
- <逻辑非表达式> → !<关系表达式> | <关系表达式>
- <关系表达式> → <算术表达式> <关系运算符> <算术表达式> | <算术表达式>
- <算术表达式> → <项> | <算术表达式> <加减运算符> <项>
- <项> → <原子表达式> | <项> <乘除运算符> <原子表达式>
- <原子表达式> → <变量标识符> | <常量标识符> | <函数调用表达式> | <类型转换表达式> | <常数> | (<表达式>)
- <类型转换表达式> → <类型>(<表达式>)
- <函数调用表达式> → <函数标识符>(<实参列表>)
- <if 语句> → if (<表达式>) <代码块> | if (<表达式>) <代码块> else <代码块>
- <while 语句> → while (<表达式>) <代码块>
- <返回语句> → return <表达式>;
- <实参列表> → ϵ | <表达式>, <实参列表>
- <类型> → Int | Float | Char | Bool
- <乘除运算符> → $*$ | $/$
- <加减运算符> → $+$ | $-$
- <关系运算符> → $==$ | $!=$ | $<$ | $<=$ | $>$ | $>=$

分析表

	类型关键字	标识符	fun	var	val	if	else	while	return	(	)	{	}	,	;	:	=
左部非终极符																	
<定义语句>				①		②		③									

左部非终极符	类型关键字	标识符	fun	var	val	if	else	while	return	(	)	{	}	,	;	:	=
<参数列表>		②									①						
<语句列表>		②	②	②	②	②		②	②			③	①				
<语句>		③		①	②	④		⑤	⑥								
<变量定义后缀>																①	②
<初始化列表后缀'>														②	①		
<初始化列表后缀'>														②		①	

变元中英文对照（AI 生成）

中文变元	英文命名建议
<程序>	Program
<定义语句列表>	DefStmtList
<定义语句列表'>	DefStmtListTail
<定义语句>	DefinitionStmt
<函数定义>	FunctionDef
<变量定义语句>	VarDefStmt
<常量定义语句>	ConstDefStmt
<参数列表>	ParamList
<代码块>	CodeBlock

中文变元	英文命名建议
<语句列表>	StmtList
<语句>	Stmt
<变量定义后缀>	VarDefSuffix
<初始化列表>	InitList
<初始化列表后缀>	InitListSuffix
<初始化列表后缀'>	InitListSuffixTail
<标识符列表后缀>	IdentListSuffix
<标识符>	Identifier
<表达式语句>	ExprStmt
<表达式>	Expression
<赋值表达式>	AssignExpr
<逻辑表达式>	LogicExpr
<逻辑或表达式>	LogicOrExpr
<逻辑与表达式>	LogicAndExpr
<逻辑非表达式>	LogicNotExpr
<关系表达式>	RelExpr
<算术表达式>	ArithExpr
<项>	Term
<原子表达式>	AtomExpr
<类型转换表达式>	TypeCastExpr
<函数调用表达式>	FuncCallExpr
<类型>	Type
<常数>	ConstValue（或 Literal）
<变量标识符>	VarIdentifier
<常量标识符>	ConstIdentifier
<函数标识符>	FuncIdentifier
<if 语句>	IfStmt
<while 语句>	WhileStmt
<返回语句>	ReturnStmt
<实参列表>	ArgList
<乘除运算符>	MulDivOp

中文变元	英文命名建议
<加减运算符>	AddSubOp
<关系运算符>	RelOp

关键字编码

id	内容
1	Int
2	Float
3	Char
4	Bool
5	Void
6	fun
7	var
8	val
9	if
10	else
11	while
12	return

界符

id	内容
1	(
2	)
3	{
4	}
5	[
6	]
7	=
8	||
9	&&
10	!
11	=
12	==

id	内容
13	!=
14	>
15	>=
16	<
17	<=
18	+
19	-
20	*
21	/
22	'
23	\
24	,
25	;
26	:

虚拟机相关

结构

- 内存：模拟一段有限，可扩展，可释放的内存（纸带）
- 寄存器：临时存储

寄存器

- BX：基址寄存器，64 位。
- DS：段寄存器，64 位。

指令

内存管理

STACK_ALLOC

语法：`STACK_ALLOC n`

作用：分配 `n` 字节空间。

STACK_FREE

语法：`STACK_FREE n`

作用：释放 `n` 字节空间。

基本内存操作

`[a, b]` 表示取内存中 `a` 开始，`b` 字节的数，默认以 `DS` 为段，默认 `b` = 4。

如 `[BX + 6, 1]` 表示取 `DS + BX + 6` 地址上的数。 特殊的，`::[BX]` 表示不使用 `DS` 为段，如 `::[2]` 表示内存中 `2` 地址上的数。

MOV

语法：`MOV dst, src`

作用：从 `src` 地址读取值，存入 `dst` 地址或寄存器。

算术与逻辑

ADD

语法：`ADD dst, a, b`

作用：`dst` = `a` + `b`，`a` 和 `b` 是地址，结果写入 `dst`。

SUB

语法：`SUB dst, a, b`

作用：减法。

MUL

语法：`MUL dst, a, b`

作用：乘法。

DIV

语法：`DIV dst, a, b`

作用：除法。

G

语法：`G dst, a, b`

作用：大于。

GE

语法：`GE dst, a, b`

作用：大于等于。

L

语法：`L dst, a, b`

作用：小于。

LE

语法：`LE dst, a, b`

作用：小于等于。

E

语法：`E dst, a, b`

作用：等于。

NE

语法：`NE dst, a, b`

作用：不等于。

AND

语法：`AND dst, a, b`

作用：与。必须为 1 字节。

OR

语法：`OR dst, a, b`

作用：或。必须为 1 字节。

NOT

语法：`NOT dst, a, b`

作用：非。必须为 1 字节。

此外还有 `ADDF`、`SUBF`、`MULF`、`DIVF` 这些用于浮点数计算，使用这些指令是对应地址必须为 8 字节，64 否则应当报错。 此外还有 `GF`、`GEF`、`LF`、`LEF`、`EF`、`NEF` 这些用于浮点数比较，使用这些指令是对应地址必须为 8 字节，64 否则应当报错。

跳转与条件控制

JMP

语法：`JMP pos`

作用：跳转到第 `pos` 条指令执行。

JNZ

语法：`JNZ addr, pos`

作用：如果 `addr` 上的值是 0，则跳转。

JZ

语法：`JZ addr, pos`

作用：如果 `addr` 上的值不是 0，则跳转。

类型转换

12 种类型转换，满足不同类型之间的转换。