

# SysYF--语言定义

---

## 1. SysYF--语言概要

SysYF--语言是课程实验要实现的编程语言，是 C 语言的一个子集。每个 SysYF--程序的源码存储在一个扩展名为 sy 的文件中。该文件中有且仅有一个名为 main 的主函数定义，还可以包含若干全局变量声明、常量声明和其他函数定义。SysYF--语言支持 **int/float 类型** 和元素为 int/float 类型以及它们对应的**一维数组类型**。其中 int 型整数为 32 位有符号数，float 为 32 位单精度浮点数；**const** 修饰符用于声明常量。

SysYF--语言本身没有提供输入/输出(I/O)的语言构造，I/O 是以运行时库方式提供，库函数可以在 SysYF--程序中的函数内调用。目前提供的库函数有 `getint()`、`getch()`、`getarray()`、`putint()`、`putch()`、`putarray()`。本次实验无需知道其具体实现，了解其名字即可(在测试程序中可能会出现库函数调用，你只需知道那是库函数调用即可)。后续实验会进一步说明。

**函数：**函数可以带参数也可以不带参数，参数的类型可以是 int/float，**但不包含数组类型**；函数可以返回 int/float 类型的值，或者不返回值(即声明为 void 类型)。当参数为 int/float 时，按值传递。函数体由若干变量声明和语句组成。

**变量/常量声明：**可以在一个变量/常量声明语句中声明多个变量或常量，声明时可以带初始化表达式。所有变量/常量要求先定义再使用。在函数外声明的为全局变量/常量，在函数内声明的为局部变量/常量。

**语句：**语句包括赋值语句、表达式语句(表达式可以为空)、语句块、if 语句、while 语句、break 语句、continue 语句、return 语句。语句块中可以包含若干变量声明和语句。

**表达式：**支持基本的算术运算 (+、-、\*、/、%)、关系运算 (==、!=、<、>、<=、>=)，没有任何逻辑运算。非 0 表示真、0 表示假，而关系运算的结果用 1 表示真、0 表示假。**算符的优先级和结合性以及计算规则与 C 语言一致。**

## 2. SysYF-- 语言的文法

SysYF--语言的文法采用扩展的 Backus 范式（EBNF，Extended Backus-Naur Form）表示，其中：

- 符号[...]表示方括号内包含的为可选项
- 符号{...}表示花括号内包含的为可重复 0 次或多次的项
- 终结符或者是由单引号括起的串，或者是 Ident、InstConst 这样的记号

SysYF--语言的文法表示如下，其中 CompUnit 为开始符号：

编译单元	CompUnit	→ [ CompUnit ] ( Decl   FuncDef )	
声明	Decl	→ ConstDecl   VarDecl	
常量声明	ConstDecl	→ <b>'const'</b> BType ConstDef { ',' ConstDef } <b>';'</b>	
基本类型	BType	→ <b>'int'</b>   <b>'float'</b>	
常数定义	ConstDef	→ <b>Ident</b> [ '[' ConstExp ']' ] <b>'='</b> ConstInitVal	
常量初值	ConstInitVal	→ ConstExp   '{' [ ConstExp { ',' ConstExp } ] '}'	
变量声明	VarDecl	→ BType VarDef { ',' VarDef } <b>';'</b>	
变量定义	VarDef	→ <b>Ident</b> [ '[' ConstExp ']' ]   <b>Ident</b> [ '[' ConstExp ']' ] <b>'='</b> InitVal	
变量初值	InitVal	→ Exp   '{' [ Exp { ',' Exp } ] '}'	
函数定义	FuncDef	→ FuncType <b>Ident</b> '(' [FuncFParams] ')' Block	
函数类型	FuncType	→ <b>'void'</b>   <b>'int'</b>   <b>'float'</b>	
函数形参表	FuncFParams	→ FuncFParam { ',' FuncFParam }	
函数形参	FuncFParam	→ BType <b>Ident</b>	
语句块	Block	→ '{' { BlockItem } '}'	
语句块项	BlockItem	→ Decl   Stmt	
语句	Stmt	→ LVal <b>'='</b> Exp <b>';'</b>   [Exp] <b>';'</b>   Block   <b>'if'</b> '(' Cond ')' Stmt [ <b>'else'</b> Stmt ]   <b>'while'</b> '(' Cond ')' Stmt   <b>'break'</b> <b>';'</b>   <b>'continue'</b> <b>';'</b>   <b>'return'</b> [Exp] <b>';'</b>	
表达式	Exp	→ AddExp	注：SysYF--表达式是 int/float 型表达式
条件表达式	Cond	→ EqExp	
左值表达式	LVal	→ <b>Ident</b> { '[' Exp ']' }	
基本表达式	PrimaryExp	→ '(' Exp ')'   LVal   Number	
数值	Number	→ <b>IntConst</b>   <b>FloatConst</b>	

一元表达式	UnaryExp	→ PrimaryExp   <b>Ident</b> '(' [FuncRParams] ')'   UnaryOp UnaryExp
单目运算符	UnaryOp	→ '+'   '-'
函数实参表	FuncRParams	→ Exp { ',' Exp }
乘除模表达式	MulExp	→ UnaryExp   MulExp ('*'   '/'   '%') UnaryExp
加减表达式	AddExp	→ MulExp   AddExp ('+'   '-') MulExp
关系表达式	RelExp	→ AddExp   RelExp ('<'   '>'   '<='   '>=') AddExp
相等性表达式	EqExp	→ RelExp   EqExp ('=='   '!=') RelExp
常量表达式	ConstExp	→ AddExp      注：使用的 Ident 必须是常量

## SysYF-- 语言的终结符特征

### 1、标识符 Ident

SysYF--语言中标识符 **Ident** 的规范如下(identifier):

标识符      identifier → identifier-nondigit  
| identifier identifier-nondigit  
| identifier digit

其中 identifier-nondigit 为下划线或以下之一

a	b	c	d	e	f	g	h	i	j	k	l	m
n	o	p	q	r	s	t	u	v	w	x	y	z
A	B	C	D	E	F	G	H	I	J	K	L	M
N	O	P	Q	R	S	T	U	V	W	X	Y	Z

digit 为以下之一

**0 1 2 3 4 5 6 7 8 9**

注：请参考 ISO/IEC 9899 <http://www.open-std.org/jtc1/sc22/wg14/www/docs/n1124.pdf>

第 51 页关于标识符的定义

同名标识符的约定：

- 全局变量和局部变量的作用域可以重叠，重叠部分局部变量优先；同名局部变量的作用域不能重叠；
- SysYF--语言中变量名可以与函数名相同。

### 2、注释

SysYF--语言中注释的规范与 C 语言一致，如下：

- 单行注释：以序列 ‘//’ 开始，直到换行符结束，不包括换行符。

- 多行注释：以序列 ‘/\*’ 开始，直到第一次出现 ‘\*/’ 时结束，包括结束处 ‘\*/’。

注：请参考 ISO/IEC 9899 <http://www.open-std.org/jtc1/sc22/wg14/www/docs/n1124.pdf> 第 66 页关于注释的定义

### 3、数值常量

SysYF--语言中数值常量可以是整型数 **IntConst**，其规范如下（对应 integer-const）：

```

整型常量      integer-const  → decimal-const | octal-const
                                   | hexadecimal-const
decimal-const  → nonzero-digit | decimal-const digit
octal-const    → 0 | octal-const octal-digit
hexadecimal-const  hexadecimal-prefix hexadecimal-digit
                                   | hexadecimal-const hexadecimal-digit

```

hexadecimal-prefix → ‘0x’ | ‘0X’

nonzero-digit 为以下之一

**1 2 3 4 5 6 7 8 9**

octal-digit 为以下之一

**0 1 2 3 4 5 6 7**

hexadecimal-digit 为以下之一

**0 1 2 3 4 5 6 7 8 9 a b c d e f A B C D E F**

注：请参考 ISO/IEC 9899 <http://www.open-std.org/jtc1/sc22/wg14/www/docs/n1124.pdf> 第 54 页关于整型常量的定义，在此基础上忽略所有后缀。

SysYF--语言中数值常量也可以是浮点型数 **FloatConst**，其定义请参考 ISO/IEC 9899 <http://www.open-std.org/jtc1/sc22/wg14/www/docs/n1124.pdf> 第 57 页关于浮点常量的定义，在此基础上忽略所有后缀。

## 3. SysYF-- 语言的语义约束

符合上述文法的程序集合是合法的 SysYF--语言程序集合的超集。下面进一步给出 SysYF--语言的语义约束。

## CompUnit

编译单元       $\text{CompUnit} \rightarrow [\text{CompUnit}] (\text{Decl} \mid \text{FuncDef})$   
声明               $\text{Decl} \rightarrow \text{ConstDecl} \mid \text{VarDecl}$

1. 一个 SysYF--程序由单个文件组成, 文件内容对应 EBNF 表示中的 **CompUnit**。在该 **CompUnit** 中, 必须存在且仅存在一个标识为 ‘main’、无参数、返回类型为 **int** 的 **FuncDef**(函数定义)。main 函数是程序的入口点, main 函数的返回结果需要输出。
2. **CompUnit** 的**顶层**变量/常量声明语句(对应 **Decl**)、函数定义(对应 **FuncDef**)都不可以重复定义同名标识符 (**Ident**), 即便标识符的类型不同也不允许。
3. **CompUnit** 的变量/常量/函数声明的作用域从该声明处开始到文件结尾。

## ConstDef

常数定义       $\text{ConstDef} \rightarrow \text{Ident} \{ '[\text{ConstExp}]' \} '=' \text{ConstInitVal}$

1. **ConstDef** 用于定义符号常量。**ConstDef** 中的 **Ident** 为常量的标识符, 在 **Ident** 后、‘=’之前是可选的数组维度的定义部分, 在 ‘=’ 之后是初始值。
2. **ConstDef** 的数组长度的定义部分不存在时, 表示定义单个变量。此时 ‘=’ 右边必须是单个初始数值。
3. **ConstDef** 的数组长度的定义部分存在时, 表示定义数组。其语义和 C 语言一维数组一致, 下标从 0 编号。 **ConstDef** 中表示数组长度的 **ConstExp** 都必须在编译时求值到**非负整数**。

ISO/IEC 9899 <http://www.open-std.org/jtc1/sc22/wg14/www/docs/n1124.pdf> 第 125 页

6.7.8 节的第 6 点规定如下:

注意: SysYF--在声明数组时各维长度都需要显式给出, 而不允许是未知的。

4. 当 **ConstDef** 定义的是数组时, ‘=’ 右边的 **ConstInitVal** 表示常量初始化器。**ConstInitVal** 中的 **ConstExp** 是能在编译时求值的 **int** 或 **float** 型表达式, 其中

6 If a designator has the form

[ *constant-expression* ]

then the current object (defined below) shall have **array** type and the expression shall be an integer constant expression. If the **array** is of unknown size, any nonnegative value is valid.

可以引用已定义的符号常量。

5. **ConstInitVal** 初始化器必须是以下三种情况之一(注: 初始值可以是 **Number**,

或者是对应类型的常量表达式):

- 一对花括号 `{}`，表示所有元素初始为 0。
- 如果花括号括起来的列表中的初始值少于数组长度，则该数组其余部分将被隐式初始化，需要被隐式初始化的元素均初始为 0。

## VarDef

**变量定义**      VarDef → **Ident** { '[' ConstExp ']' }  
                     | **Ident** { '[' ConstExp ']' } '=' InitVal

1. **VarDef** 用于定义变量。当不含有‘=’和初始值时，其运行时实际初值未定义。
2. **VarDef** 的数组长度的定义部分不存在时，表示定义单个变量。存在时，和 **ConstDef** 类似，表示定义数组。（参见 **ConstDef** 的第 2 点）
3. 当 **VarDef** 含有 ‘=’ 和初始值时，‘=’ 右边的 **InitVal** 和 **CostInitVal** 的结构要求相同，唯一的区别是 **ConstInitVal** 中的表达式是 **ConstExp** 常量表达式，而 **InitVal** 中的表达式可以是当前上下文合法的任何 **Exp**。
4. **VarDef** 中表示数组长度的 **ConstExp** 必须是能求值到**非负整数**，但 **InitVal** 中的初始值为 **Exp**，其中可以引用变量。

初值

常量初值	ConstInitVal	→ ConstExp
		'{' [ ConstExp {',' ConstExp } ] '}'
变量初值	InitVal	→ Exp   '{' [ Exp {',' Exp } ] '}'

1. 全局变量声明中指定的初值表达式必须是常量表达式。
2. 常量或变量声明中指定的初值要与该常量或变量的类型一致

如下形式的 VarDef / ConstDef 不满足 SysYF--语义约束:

```
a[4] = 4
a[2] = {1, 2, 3}
a = {1, 2, 3}
```

3. 未显式初始化的局部变量，其值是不确定的；而未显式初始化的全局变量，其（元素）值均被初始化为 0。

## FuncFParam 与实参

函数形参       $\text{FuncFParam} \rightarrow \text{BType Ident}$

1. FuncFParam 定义一个函数的一个形式参数。
2. 函数实参的语法是 Exp。对于 int/float 类型的参数，遵循按值传递。

## FuncDef

函数定义       $\text{FuncDef} \rightarrow \text{FuncType Ident '([FuncFParams])' Block}$

1. FuncDef 表示函数定义。其中的 FuncType 指明返回类型。
  - a) 当返回类型为 int/float 时，函数内所有分支都应当含有带有 Exp 的 return 语句。不含有 return 语句的分支的返回值未定义。
  - b) 当返回值类型为 void 时，函数内只能出现不带返回值的 return 语句。
2. FuncDef 中形参列表 (FuncFParams) 的每个形参声明 (FuncFParam) 用于声明 int/float 类型的参数。FuncFParam 的语义参见前文。

## Block

1. Block 表示语句块。语句块会创建作用域，语句块内声明的变量的生存期在该语句块内。
2. 语句块内可以再次定义与语句块外同名的变量或常量（通过 Decl 语句），其作用域从定义处开始到该语句块尾结束，它隐藏语句块外的同名变量或常量。

## Stmt

1. Stmt 中的 if 类型语句遵循就近匹配。
2. 单个 Exp 可以作为 Stmt。Exp 会被求值，所求的值会被丢弃。

## LVal

1. LVal 表示具有左值的表达式，可以为变量或者某个数组元素。
2. 当 LVal 表示数组时，方括号个数必须和数组变量的维数相同（即定位到元素）。

3. 当 LVal 表示单个变量时，不能出现后面的方括号。

## Exp 与 Cond

1. Exp 在 SysYF--中代表 int/float 型表达式，故它定义为 AddExp；Cond 代表无逻辑运算的关系表达式，故它定义为 EqExp。
2. LVal 必须是当前作用域内、该 Exp 语句之前有定义的变量或常量；对于赋值号左边的 LVal 必须是变量。
3. 函数调用形式是 Ident ‘(’ FuncRParams ‘)’, 其中的 FuncRParams 表示实际参数。实际参数的类型和个数必须与 Ident 对应的函数定义的形参完全匹配。
4. SysYF--中算符的优先级与结合性与 C 语言一致，在上面的 SysYF--文法中已体现出优先级与结合性的定义。