

# Mx\* Language Reference Manual

2016 年 4 月 4 日

## 1 程序结构

Mx\* 语言由以下要素构成

- 函数定义 (function definition)
- 类定义 (class definition)
- 接口定义 (interface definition)
- main 函数是 Mx\* 程序的顶层结构 (toplevel) 的一部分。程序从 main 函数开始执行。main 函数没有参数，返回值为整数。
- 全局变量声明。
- 源程序大小不超过 16M

例 1:

```
int main() {  
    println("too young too simple.");  
    return 0;  
}
```

例 2:

```
// 函数调用在顶层结构后  
int main() {  
    int i;  
    for (i = 0; i < 3; i++)  
        angry();  
}
```

```
        return 0;  
    }  
    void angry() {  
        print("I'm angry!\n");  
    }
```

例 3:

```
int Wallace = 1 << 10;  
class sometimes {  
    int naive;  
}  
int main() {  
    sometimes keep = new sometimes;  
    keep.naive = 0;  
    while (getInt() < Wallace) {  
        keep.naive++;  
    }  
    return 0;  
}
```

## 2 语法规则

### 2.1 源文件编码

ASCII 编码，区分大小写，不接受中文字符。

## 2.2 关键字 Reserved Keywords

```
bool int string null void
true false
if for while
break continue return
new class
```

## 2.3 空白字符的处理

空格、制表符、回车符和换行符在源文件中除了区分词素 (Token) 外没有其他含义。

## 2.4 注释

从// 开始到本行结束的内容都会被作为注释

## 2.5 标识符

标识符的第一个字符必须是英文字母，第二个字符开始可以是英文字母、数字或者下划线。标识符区分大小写，长度不超过 64 个字符。

## 2.6 常量

### 2.6.1 逻辑常量

true 为真，false 为假

### 2.6.2 整数常量

整数常量以十进制表示。整数常量不设负数，负数可以由正数取负号得到。编译器至少应该能处理大小范围在  $[-2^{31}, 2^{31})$  内的整数。

整数常量如果首位为 0，或者大小超过编译器能接受的范围，行为是未定义的。

### 2.6.3 字符串常量

字符串常量是由双引号括起来的字符串。字符串长度最小为 0，至少可以达到 255。字符串中的所有字符必须是可示字符 (printable character)，空格或者转义字符中的一种。转义字符有三个：\n 表示换行符，\\ 表示反斜杠，\" 表示双引号。

### 2.6.4 空值常量

null 用来表示引用类型没有指向任何值。

### 2.6.5 数组常量

## 3 运算符

### 3.1 算术运算符

+ - \* / %

### 3.2 关系运算符

< > == != >= <=

### 3.3 逻辑运算符

&& || !

### 3.4 位运算符

<< >> ~ | ^ &

定义右移为算术右移。例如

11100011 >> 3 == 11111100

### 3.5 赋值运算符

=

其他类似于 += 的 augmented assignment 不保留了。

## 3.6 自增运算符和自减运算符

`++ --`

## 3.7 分量运算符

`.`

## 3.8 下标运算符

`[]`

## 3.9 括号

`()`

圆括号可以用于 calling functions 和 subexpression grouping。

## 3.10 优先级

和 C 语言一致。运算符的优先级从高到低大致是：单目运算符、算术运算符、关系运算符、逻辑运算符、条件运算符、赋值运算符。

# 4 数据类型

## 4.1 基础类型

**bool 类型** 略

**int 类型** 略

**void 类型** void 类型是用来表示函数没有返回值的特殊类型。只能在定义函数的返回值类型时使用。如果想说明一个函数没有参数，不必写 void，直接让参数列表为空即可。

**string 类型** 字符串类型属于引用类型。字符串本身不能改变 (immutable)。

## 4.2 复合类型

### 4.2.1 数组

数组是可以动态创建的引用类型，长度无需在声明时确定。

```
string[] vec;  
vec = new string[10];
```

注意 java 声明数组时，既可以写 `int[] a`，也可以写 `int a[]`，我们不支持后者的书写习惯。

### 4.2.2 数组的内建方法

```
int size()
```

该方法可以返回数组的长度。如果数组为 null，结果是未定义的。

### 4.2.3 交错数组

我们使用交错数组 (Jagged Array) 来达到多维数组的效果。交错数组就是数组的数组。交错数组的申明方法和 C# 保持一致。

```
int[][] matrix;
```

交错数组的创建语句如下：

```
int[][] graph = new int[3][];  
graph[0] = null;  
graph[1] = new int[10];  
graph[2] = new int[30];
```

需要先创建最外层数组的空间，然后再创建内层数组的空间。

### 4.2.4 交错数组的文法糖

```
int[][] matrix = new int[3][4];
```

这个文法糖产生的效果是未定义的。支持交错数组的主流语言中，C# 不支持这么做，Java 支持。建议学生支持一下，如果测试数据比较多地用到多维数组，能让测试数据简洁不少。

## 5 类

类的定义通过以下形式

```
class 类名 {  
    类型1 字段名1;  
    类型2 字段名2;  
}
```

暂时不支持自定义方法，也不支持 `private` 修饰符，也不支持继承、抽象类或接口。没有构造函数，没有成员的默认初始化语句，没有析构函数。

## 6 表达式

### 6.1 单目表达式

单目表达式有常量，标识符变量名。脑补中

### 6.2 双目表达式

脑补中

## 7 语句

### 7.1 声明语句

类型 变量名;

或者

类型 变量名 = 初始表达式;

### 7.2 表达式语句

表达式;

### 7.3 条件语句

```
if (表达式1)  
    语句1  
else if (表达式2)  
    语句2  
else  
    语句3
```

### 7.4 循环语句

```
while (表达式)  
    语句
```

或者

```
for (表达式1;表达式2;表达式3)  
    语句
```

### 7.5 跳转语句

```
return 表达式;  
break;  
continue;
```

## 8 函数

### 8.1 函数定义

```
类型 函数名 (参数序列) {  
}
```

Mx\* 不支持声明函数的签名，也不支持在一个函数内嵌套申明另一个子函数或类。

## 8.2 内建函数

内建函数是指系统直接提供给用户的函数，不需要申明就可以使用。

```
void print(string str);
```

向标准输出流中输出字符串 `str`。

```
void println(string str);
```

向标准输出流中输出字符串 `str`，并在结尾处加上换行符。

```
string getString();
```

从标准输入流里读取一行字符并返回。

```
int getInt();
```

从标准输入流里读取一个整数并返回，如果输入流里并不是一个合法的整数，结果是未定义的。

```
string toString(int i);
```

将一个整数转化为字符串。

## 9 字符串

```
// 错误！字符串不能赋值null
```

```
string str = null;
```

```
// 正确
```

```
string[] str_arr = null;
```

### 9.1 运算符语义

- `+` 表示两个字符串的拼接
- `==` 比较的是两个字符串内容是否完全一致，而不是比较内存地址
- `<` 比较字典序大小，其余关系运算符同理
- 其他运算符为非法。

### 9.2 字符串的内建方法

```
int length()
```

返回字符串长度。

```
string substring(int left, int right)
```

返回下标从 `left` 开始到 `right` 结束的子串。

```
int parseInt();
```

如果字符串的前缀是一个整数，则返回这个整数，不然结果未定义。

```
int ord(int pos);
```

返回字符串中的第 `pos` 位上的字符的 ASCII 码。位置从 0 开始编号。

### 9.3 字符串常数的内建方法

形如

```
"Four score and seven years ago".length();
```

这样的表达式，所产生的效果是未定义的。编译器可以报错，也可以返回该字符串常数的长度，学生可以自由选择实现。

## 10 null 的用法

`null` 表示数组或者某个对象为空，不能用在 `int`，`bool` 和 `string` 上：如果数组已经为空，再引用其某个下标，结果是未定义的。

## 11 左值表达式

以下表达式称为左值表达式：

- 单个变量标识符

- 形如 `xxx[yyy]` 的表达式，其中 `xxx` 必须是左值
- 形如 `xxx.yyy` 的表达式，其中 `xxx` 必须是左值

函数调用表达式不算左值表达式，比如 `f().member` 不能出现在赋值运算符的左侧。

## 12 作用域规则 Scope Rule

一个符号起作用的那一段程序区域称为这个变量的作用域。

- 在一段语句中，由 `{` 和 `}` 组成的块会引进一个新的作用域
- 用户定义函数入口会引入一个新的作用域
- 用户定义类的入口会引入一个新的作用域
- 全局变量和局部变量不支持前向引用，作用域为声明开始的位置直到最近的一个块的结束位置
- 函数和类的声明都应该在顶层，作用域为全局，支持前向引用

## 13 命名空间

所有符号共享一个命名空间，所以变量不能和函数或类同名，不允许任何函数重载。