Mx* Language Reference Manual

2016 年 4 月 2 日

1 程序结构

Mx* 语言由以下要素构成

- 函数定义（function definition）定义了函数。函数允许回溯引用，所以其定义位置既可以在调用之前，也可以在调用之后。
- 类定义（class definition）和函数一样，允许回溯引用。
- ~~接口定义 (interface definition)~~
- main 函数是 Mx* 程序的顶层结构 (toplevel)。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。全局变量不支持回溯引用
- 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() {
  println("I'm angry!");
}
```

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 文法规则

2.1 源文件编码

ACSII 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 $[-2^{15}, 2^{15})$ 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[]，我们不允许按后者书写。

4.2.2 数组的内建方法

```
int size()
```

该方法可以返回数组的大小长度。

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];
```

以上代码一维接一维声明了一个二维数组，这是和 Java 或者 C# 保持一致的。

4.2.4 交错数组的文法糖

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

我正在纠结要不要引入这个文法糖。C# 不支持，Java 是支持的。暂时先不支持吧。我们看一下测试数据，如果比较多地用到多维数组，再把它加回来，毕竟能让代码简洁不少。

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

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* 不支持没有函数体的签名声明。也不支持函数内嵌套申明函数或类。

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。

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。