Mx* Language Reference Manual

2017 年 4 月 3 日

1 用词说明

未定义（Undefined Behavior）—~~指中央还没有表态~~ 指规范并没有定义该情况发生时语言的表现。初衷是为了给同学们提供一些自己发挥的空间，在测试数据里，这些没有定义的情况是不会发生的。例：

- 术语：编译器接受源代码长度如果超过 1M，结果是未定义的
- 解释：我们测试用的源代码长度不会超过 1M，可以假设测试数据不会出现这种情况。如果学生乐意，也可以让自己的编译器支持比 1M 更大的文件，或者也可以在源代码超过这个长度之后报错，拒绝编译。

总之，“未定义”=“这种情况在测试数据里不会发生”。所以在提交新测试数据的时候，也要注意不能出现有未定义情况的程序。

2 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）
- main 函数是 Mx* 程序的顶层结构 (toplevel) 的一部分。程序先按照顺序初始化所有全局变量，然后从 main 函数开始执行。main 函数没有参数，返回值为整数。一个程序不可以没有 main 函数。
- 全局变量声明
- 接口定义 (interface definition) 是未定义的

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;
  void make_money() {
     this.naive++;
  }
}
int main() {
  sometimes keep = new sometimes;
  keep.naive = 0;
  while (getInt() < Wallace) {
    keep.make_money();
  }
  return 0;
}
```

3 文法规则

3.1 源文件编码

ACSII 编码，区分大小写。中文字符是未定义的。

3.2 关键字 Reserved Keywords

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

3.3 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。

3.4 注释

从// 开始到本行结束的内容都会被作为注释。类似于/**/的注释是未定义的。

3.5 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。长度超过 64 个字符的标识符是未定义的。

3.6 常量

3.6.1 逻辑常量

true 为真，false 为假

3.6.2 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。

首位为 0 的整数常量是未定义的，大小超过上述范围的整数是未定义的。

3.6.3 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，长度超过 255 的字符串是未定义的。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。其余出现在 C++ 语言里的转义字符是未定义的。

3.6.4 空值常量

null 用来表示引用类型没有指向任何值。

3.6.5 数组常量

未定义

4 运算符

4.1 算术运算符

`+ - * / %`

4.2 关系运算符

`< > == != >= <=`

4.3 逻辑运算符

`&& || !`

4.4 位运算符

`<< >> ~ | ^ &`

定义右移为算术右移。例如

```
11100011 >> 3 == 11111100
```

4.5 赋值运算符

`=`

赋值运算符的结果是未定义的。之所以这么做，是方便大家能用两种常见的方法处理赋值运算符：或者可以像 C 语言一样支持连续赋值，比如 a=b=c；也可以定义赋值运算符的结果类型是 void，这样可以避免写出 if (a=b) 这样的代码。其他类似于 += 的 augmented assignment 的运算是未定义的。

4.6 自增运算符和自减运算符

`++ --`

4.7 分量运算符

.

4.8 下标运算符

`[]`

4.9 括号

`()`

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

4.10 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。

5 数据类型

5.1 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）。

5.2 复合类型

5.2.1 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。数组长度超过 $2^{31}-1$ 是未定义的。

```
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[]，但我们不支持后者。

5.2.2 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。

5.2.3 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。

5.2.4 方便声明多维数组的文法糖

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

在 14 级里，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。为了方便，从 15 级开始，我们就支持这种文法糖吧。

6 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;
  类型3 函数名(参数序列) {
    各类语句
  }
  类名() { // 可有可无的构造函数
    各类语句
  }
}
```

6.1 用户自定义类的方法

类似于函数。除了构造函数之外，都有返回值或 void 修饰符。

6.2 构造函数

构造函数是可选的。类定义中可以没有构造函数，此时类的初始化行为是未定义的。

由于函数和类的重载是未定义的，所以，出现在标准测试集里的程序，每个类最多应该只有一种构造函数，而且这个函数不带参数。如果想实现构造函数重载，请尽管干吧。

6.3 未定义的东西

以下面向对象的基本特性都是未定义的

- private 修饰符
- 继承
- 抽象类或接口
- 多态
- 成员的默认初始化表达式
- 析构函数

7 表达式

7.1 单目表达式

单目表达式有常量，标识符变量名。等等

7.2 双目表达式

似乎不需要定义得太详细，大家都懂的。

8 语句

8.1 声明语句

```
类型 变量名;
```

或者

```
类型 变量名 = 初始表达式;
```

Java 约定，如果没有初始表达式，则变量的初始值为 0 或 null，我们对此类行为没有定义。也就是说，标准测试集里的程序，应该保证变量在使用前先被赋值了，不然就不是一个合格的测试点。

8.2 表达式语句

```
表达式;
```

8.3 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
```

判断表达式不能为空，且类型必须为 bool

8.4 循环语句

```
while (表达式)
  语句
```

while 的判断表达式不能为空，且类型必须为 bool

```
for (表达式1;表达式2;表达式3)
  语句
```

for 语句的表达式比较特殊，判断表达式可以为空，但如果不空的情况下，类型必须为 bool

8.5 跳转语句

```
return 表达式;
break;
continue;
```

9 函数

9.1 函数定义

```
类型 函数名 (参数序列) {
}
```

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

9.2 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。

10 null

null 表示数组或者某个对象为空，不能用在 int，bool 上。如果数组为 null，再引用其某个下标，结果是未定义的。string 也不能赋值为 null——虽然 c# 和 java 是允许这么做的，但这样会不得不说明 null 和其他字符串的运算规定，这些叙述太复杂了。

11 字符串

```
// 错误！字符串不能赋值null
string str = null;
// 正确
string[] str_arr = null;
```

11.1 涉及字符串的运算符的语义

- + 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（结构相等），而不是比较内存地址（引用相等）
- < 比较字典序大小，其余关系运算符同理
- 其他运算符的表现是未定义的

11.2 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();`

返回一个整数，这个整数应该是该字符串的最长前缀。如果该字符串没有一个前缀是整数，结果未定义。如果该整数超界，结果也未定义。

`int ord(int pos);`

返回字符串中的第 pos 位上的字符的 ASCII 码。下标从 0 开始编号。

11.3 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。大家可以认为这是个语法错误，或者也可以返回该字符串常数的长度。

12 左值的定义

关于左值的定义，如果直接继承 C 语言对于左值的定义，会有一些问题（因为 C 语言的结构类型是可以存在栈里的。所以直接复制会带来歧义）。

我们参考的是 Java 的官方说明手册。`https://docs.oracle.com/javase/specs/jls/se7/html/jls-4.html#jls-4.12.3` 首先，Java 的左值不称作左值，而是称为 variable（见该文档 15.1. Evaluation, Denotation, and Result 的注释）。简单来说 Java 的左值可以是以下几种:

- Method parameters name argument values passed to a method（函数的形参）
- Local variables are declared by local variable declaration statements（局部变量）
- An instance variable is a field declared within a class declaration（类的一个成员）
- Array components are unnamed variables that are created and initialized to default values whenever a new object that is an array is created （数组的一个元素）

我们要求语言至少能支持上述四种类型作为左值。C++ 支持更多的左值形式，而我们的语言对更多类型的左值是未定义的。

13 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，该作用域里申明的所用成员，作用域为整个类。
- 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（forward reference）。

14 命名空间

所有符号共享一个命名空间，所以在同一个作用域里，变量，函数，和 class，都不能同名。不同作用域的时候，内层作用域可以遮蔽外层作用域的名字。

函数重载或类的重载，都是未定义的。