PLANG 用户文档（第一版）

Plang 是一种采用面向对象设计思想的高级程序设计语言。该语言主要用于 Microsoft Windows10 平台下小型 VC++系统的嵌入式脚本开发。本文档通过简单的示例让大家了解 Plang 编程语言。

我的第一个 PLANG 程序

```
import std::IO;
public class HelloWorld {
    public static main(){
        IO.printf("Hello, world!");
    }
}
```

运行：`plang -c HelloWorld -s 所在目录`

输出：

```
$ plang -c HelloWorld -s src
Hello, world!
```

命令解析：

以上使用了 plang 命令。其中 `plang -c HelloWorld` 指明运行的类名；而 `plang -s src` 指明源文件路径。

PLANG 介绍

- Plang 语言是简单的：
  - 继承 C++、Java 语言的语法特点。
  - 消除 void、extends、implements 等关键字。

- 支持简单的内存自动回收功能。

- Plang 语言是面向对象的：
  - 提供类、接口和继承等面向对象特性。
- Plang 语言是解释型的：
  - Plang 程序(后缀为 ps 的文件)被编译成字节码格式(后缀为 p 的文件)，然后可以在 Win10 32/64 位平台下由解释器运行。
- Plang 语言是支持协程的：
  - Plang 语言在语法层面支持协程。协程是一种单线程环境下支持异步执行的解决方案。相比内核调度的多线程，协程将调度权交给开发者，以减少内核环境切换的开销，更能适应大量且耗时极短的异步应用场景。

Plang 开发工具暂无，仅提供 dos 风格的解释器、虚拟机和调试器。

PLANG 基础语法

一个 Plang 程序可以认为是一系列的对象的集合，这些对象通过调用彼此的方法来协同工作。下面简要介绍类、对象、方法、字段的概念。

- 类：类是一个模板，描述一类对象的行为和状态。
- 对象：对象是类的实例，具有行为和状态。
- 方法：方法是对象行为的定义。
- 字段：字段是用来存储对象状态的。

基本语法

- 大小写敏感。
- 类名：属于标识符的一种，建议大写字母开头。
- 方法名：属于标识符的一种，建议小写动词开头。
- 包名：属于标识符的一种，建议小写；
- 源文件名：保证不重复，并且后缀规定为.ps。
- 主方法：作为独立的编程语言来讲，Plang 程序由 public static main()方

法开始执行；作为嵌入式语言来讲，任何静态非本地方法都能作为方法入口。

标识符

Plang 所有的组成部分都需要名字。类名、变量名、方法名都被称为标识符。

沿袭 C 语言标识符格式：

- 所有标识符以字母（A-Z 或 a-z）、下划线（_）开始。
- 首字符后可以是字母（A-Z 或 a-z）、下划线（_）、数字的任何组合。
- 合法标识符举例：Hello、abc、_boy、bug_1_2。
- 非法标识符举例：123abc、&me。

修饰符

- 访问控制符：public、protected、private、friend；
- 非访问控制符：static、native 等；

变量

- 局部变量
- 静态变量
- 非静态变量

Plang 关键字

类型	关键字	说明
访问控制	private	私有的
	protected	受保护的

	public	公共的
	friend	友元的
非访问控制	class	类
	interface	接口
	native	本地，原生方法
	new	创建
	static	静态
	const	常量
程序控制	break	跳出循环
	continue	继续
	do	运行
	else	否则
	if	如果
	instanceof	实例
	return	返回
	while	循环
	co_start	分发协程
	co_await	阻塞协程
	co_yield	让出、唤醒协程
错误处理	catch	捕捉异常
	throw	抛出一个异常对象
	try	捕获异常
基本类型	boolean	布尔型
	byte	字节型
	char	字符型
	double	双精度浮点
	float	单精度浮点
	int	整型
	long	长整型

	short	短整型
变量引用	super	父类、超类
	this	本类

Plang 注释

类似于 C/C++、Plang 也支持单行以及多行注释。

示例：

```
//单行注释；
/*
多行注释
*/
```

继承

- 一个类可以继承另一个类。子类继承父类公共、受保护的非静态字段与方法。
- 一个类可以实现多个接口。实现类必须定义所有接口方法。

示例：

```
class A{ }
interface IA{ }
interface IB{ }
class B : A, @IA, @IB{ }
```

友元类

- 在一个类中通过使用 friend 关键字，声明友元类。
- 一个友元类允许访问该类下的 protected、private 成员。
- 一个友元类无法访问该类的父类或子类的 protected、private 成员。

示例：

```
class Friend
{
    a->A=null;
    public Friend()
    {
        //访问 A 类私有成员；
        temp->int=new A().data;
    }
}

class A
{
        friend class Friend;
        private data->int=0;

        public A(){ }
        public static main(){
            f->Friend=new Friend();
        }
}
```

PLANG 对象和类

Plang 作为一种面向对象语言，支持以下基本概念：

- ✓ 多态
- ✓ 继承
- ✓ 封装
- ✓ 抽象
- ✓ 类

- ✓ 对象
- ✓ 方法
- ✓ 重载

- Plang 中的类

  类的默认访问修饰符为 protected，即允许包内访问。类成员的默认访问修饰符为 private，即允许类内访问。

  示例：

```
public class Man
{
    protected name->const char[];
    protected age->int;

    public Man(){}
    public eat()->int{}
    public walk(){}
    public sleep(){}
}
```

- Plang 中的接口

  接口的默认访问修饰符为 protected，即允许包内访问。接口方法的默认访问修饰符为 public，因此无需添加访问修饰符。

  示例：

```
public interface JDBC
{
        select(statement->Statement)->ResultSet;
}
```

- 构造方法

每个类可以有构造方法。如果不提供构造方法，则无法创建对象。另外子类可以调用父类构造方法。注意：构造方法名称必须与类名一致，并且返回类型为空。

示例：

```
public class John : Man
{
    public John(){ super(); }
    public John(name->const char[], age->int){ super(); }
}
```

- 创建对象

类似 Java，使用 new 关键字创建对象。

示例：

```
public class John : Man
{
    public John(){ }
    public static main()
    {
        //诞生一个 john；
        john->John=new John();
    }
}
```

- 访问字段和方法

示例：

```
//静态访问（[包名::]类名.成员）；
my::John.main();
//非静态访问（对象.成员）；
john->John=new John();
```

```
//访问字段;
john.name;
//访问方法;
john.eat();
```

- 源文件格式
  - 一个源文件可以包含多个 public 类。
  - 同一个包中不能出现类名相同的类。
  - 如果一个类定义在某个包中，那么 package 语句应该放在源文件首行。
  - 如果源文件中包含 import 语句，那么应该放在 package 语句和类定义之间。如果没有 package 语句应该放在源文件最前面。
  - import 语句和 package 语句对源文件中定义的所有类都有效。在同一源文件中，不能给不同的类、不同的包声明。

  示例：

```
//声明包（可选）
package PackName;
//引入类（可选）
import PackName::ClassName;
//声明类（可选）
public class ClassName{}
```

- Import 语句

  Import 语句就是用来提供一个合理的路径，方便编译器可以找到某类。

PLANG 基本数据类型

Plang 语言提供基本数据类型、引用类型、常量。

- 基本数据类型

数据类型	最小值	最大值

char	-128（-2^7）	127（2^7-1）
byte	0	256（2^8）
short	-32768（-2^15）	32767（2^15 - 1）
int	-2,147,483,648（-2^31）	2,147,483,647（2^31 - 1）
long	-9,223,372,036,854,775,808（-2^63）	9,223,372,036,854,775,807（2^63 -1）
float	IEEE 754 标准	IEEE 754 标准
double	IEEE 754 标准	IEEE 754 标准
boolean	false	true

示例：

```
public static main()
{
        i->int=0;
        l->long=0L;
        f->float=0.0F;
        d->double=0.0D;
        c->char='a';
        b->boolean=false;
}
```

- 引用类型
  - 对象、数组都属于引用类型。
  - 引用类型默认值为 null。
  - 一个引用变量可以引用任何兼容的类型。

- 常量
  - 常量不允许二次修改。
  - 常量用关键字 const 声明。例如：`a->const int=0;b->const char[]=null;`

- 自动类型转换

  基本类型数据：byte,char,short<=>int<=>long<=>float<=>double。

  引用类型：根据继承关系自动转换。

PLANG 修饰符

- 访问修饰符
  - 类访问修饰符有 public，默认为 protected。
  - 类成员访问修饰符有 public，protected，private，默认为 private。
- 非访问修饰符
  - static 修饰符用来声明静态字段、静态方法。
  - const 修饰符用来声明常量。
  - native 修饰符用来声明本地方法。
  - 其他。

PLANG 运算符

- 算术运算符：+，-，*，/，%，++（前缀、后缀），--（前缀、后缀）。
- 关系运算符：==,!=,>,<,>=,<=,instanceof
- 位运算符：&,|,^,~,<<,>>,>>>
- 逻辑运算符：&&,||,!
- 赋值运算符：=,+=,-=,*=,/=,%=,<<=,>>=,&=,^=,|=
- 三目运算符：?:
- 其他运算符：(),[],.(点运算符),::（包引用符）,->（类型声明符）

PLANG 控制语句

- 循环结构

- while 循环。
- do…while 循环。
- break 中断。
- continue 中断。

➢ 条件语句

- if…else 语句
- if…else if…else 语句
- 嵌套 if…else 语句

➢ 异常处理语句

- throw 语句
- try … catch(…)语句

  示例：

```
try{
    throw new Class();
}
catch(e->ClassA){}
catch(e->ClassB){}
```

PLANG 协程调度

Plang 语言支持协程，由关键字 `co_start，co_await，co_yield` 提供支持。主方法为默认主协程，子协程的执行顺序按照分发先后依次执行。Plang 虚拟机维护一个协程队列，一个协程需要在前个协程让出执行权后，才能运行。如果程序结束时仍存在阻塞协程（程序处于死锁状态），则中断运行并且提示错误（该情况往往归咎于程序设计者）。

➢ co_start 语句

- 用于分发协程。协程需要具备以下条件：

  1. 返回值为空。
  2. 非构造方法。

3. 非 native 方法。

示例：

```
public class Foo
{
   static foo(){}
   public static main()
   {
      co_start Foo.foo();
   }
}
```

- co_await 语句
  - 用于阻塞当前协程。语句后接绑定对象实例。例如：`co_await john;`
  - 所接对象若为 null，程序报错。

示例：

```
public class Foo{
   static foo(){}
   public static main()
   {
      john->John=new John();
      co_await john;
   }
}
```

- co_yield 语句
  - co_yield 用于让出程序执行权，其后接绑定对象；
  - 若后接对象存在与之绑定的阻塞协程，则让出当前协程，并唤醒阻塞协程。例如：`co_yield john;`
  - 若后接对象为 null，则仅让出当前执行权。例如：`co_yield null;`

示例：

```
import std::IO;


class Foo {
    static foo(){
        IO.printf("in foo0\n");
        co_yield null;
        IO.printf("in foo1\n");
    }

    public static main() {
        co_start Foo.foo();
        IO.printf("in main0\n");
        co_yield null;
        IO.printf("in main1\n");
    }
}
```

输出：

```
$ plang -c Foo
in main0
in foo0
in main1
in foo1
```

PLANG 标准库

Plang 语言目前处于标准库研发阶段，目前支持的标准库有：

- std::io 输入输出库
- std::math 数学库
- std::os 系统相关库

- std::time 时间库
- std::data 数据处理库
- std::collection 集合库
- std::dom XML 库