



**软件工程专业项目设计（论文）**

**少儿编程平台**

**学 院 计算机学院**

**专 业 软件工程**

**年级班别 2016级（3）班**

**学 号 3116004824**

**学生姓名 林楷羽**

**指导教师 欧毓毅**

**2019年11月**

**目 录**

[1. 需求分析 1](#_Toc25078320)

[1.1 设计编程语言语法 1](#_Toc25078321)

[1.1.1 目标用户 1](#_Toc25078322)

[1.1.2 A语言语法 2](#_Toc25078323)

[1.2 编译器：词法分析、语法分析 5](#_Toc25078324)

[1.2.1 词法分析 5](#_Toc25078325)

[1.2.2 语法分析 6](#_Toc25078326)

[1.2.3 程序执行器 9](#_Toc25078327)

[1.3 编程平台设计：编程关卡 10](#_Toc25078328)

[1.3.1 基础阶段 11](#_Toc25078329)

[1.3.2 循环、条件判断阶段 13](#_Toc25078330)

[1.3.3 函数声明、调用阶段 17](#_Toc25078331)

[1.5 本章小结 18](#_Toc25078332)

# 需求分析

使用前端MVVM框架REACT，结合编译原理等理论知识，设计实现一款少儿编程平台：以游戏为媒介，吸引目标用户，提高对编程的兴趣，并在编程关卡中结合编程各阶段思想，如顺序执行、条件执行、循环执行、函数调用等，循序渐近地引导用户，在通关关卡的同时，掌握编程核心思想，感受编程的独特魅力。

根据以上场景进行需求分析，可以得出，本编程平台需求大致分为以下几点：

1. 设计编程语言语法

2. 编译器实现：词法分析、语法分析、程序执行

3. 编程平台设计：编程游戏区，编程用户交互区，编程关卡设计

## 1.1 设计编程语言语法

### 1.1.1 目标用户

该少儿编程平台的目标用户为青少年儿童，在此阶段的目标用户大部分心智发育已到一定阶段，但未完全发育，较为严肃深刻的教学方式难以达到预计效果。因此，需要以偏趣味性的方式去吸引目标用户，所以，本编程平台选择将游戏与教学结合，以游戏的方式去引导目标用户。

同时，业界流行的C、C++、java、go等编程语言在一定程度上较为复杂，目标用户理解并使用上有一定的困难，因此，需要设计一套更为简易的编程语言（以下简称A语言）用于本编程平台：

1. 需要以C语言等业界基础语言为原型，让的目标用户在熟练掌握本A语言后可以更加平滑顺畅的理解C语言等主流编程语言。

2. 保留C语言的基础语法设计，但去除较为艰深抽象的指针，结构体等内容，降低A语言掌握难度，便于目标用户学习与使用。

3. A语言需要实现：类型变量、顺序执行、条件执行、循环执行、函数调用等编程核心。

### 1.1.2 A语言语法

1. 数据类型：布尔类型、整形类型、字符串类型。

1）布尔类型表示一个逻辑实体，可以有两个值： true false

布尔类型支持取反操作 以及 == 比较运算

!true // false

!false // true

False == true // false

True == true // true

2） 整型表示一个整数，在程序中作为一个表达式值，可以进行运算。

如：1 2 3 23 433

整形类型支持常规的+ - \* / 运算 以及== != > < 比较运算

1 > 2 // false

1 != 2 // true

1 + 1 (1 + 2) \* 3 / 4 – 5

整形类型支持取负操作

-1 -2 –(1+2)

整形类型支持取反操作，除0外 其余逻辑实体都为true

!1 // false

!0 // true

3）字符串类型表示文本数据，在程序中作为一个表达式值。如：

“aaa”, “bbb”

字符串支持 == != 操作

“aaa” == “aaa” // true

“aaa” != “bbb” // true

字符串支持取反操作，除空字符串(“”)外，其余逻辑实体都为true

!“aaa” // false

2. 变量声明语句与变量赋值语句

使用关键字 let 与 = 可以为变量进行声明与赋值，如下

let a = 0; // 声明并赋值语句

a = 1; // 赋值语句



3. 条件判断语句。

条件判断语句指的是根据指定的条件所返回的结果（真或假或其它预定义的），来执行特定的语句。支持的条件判断语句：if...else，当一个逻辑条件为真，用if语句执行一个语句。当这个条件为假，使用可选择的 else 从句来执行这个语句。

If-else语句如下所示：

if (true) {

return 1;

} else {

return 0;

}

Else语句可省略

if (true) {

return 1;

}



4. 循环语句

循环提供了一种快速和简单的方式去做一些重复的事。

支持的条件判断语句：while，一个 while 语句只要指定的条件求值为真（true）就会一直执行它的语句块。一个 while 语句看起来像这样：

let a = 0;

while(a < 5) {

// 赋值语句

a = a + 1;

}



5. 函数定义以及函数调用

函数是一组执行任务或计算值的语句。

定义一个函数表达式需要使用一系列的fn关键字，函数参数列表（包围在小括号中，并用逗号就分隔），定义代码执行的语句（包围在大括号中）。

一个函数声明以及定义如下：

let add = fn(a, b) {

// return 语句

return a + b;

}

定义一个函数并不会自动的执行它。定义了函数仅仅是赋予函数以名称并明确函数被调用时该做些什么。调用函数才会以给定的参数真正执行这些动作。

在对函数进行声明并赋值给某个变量后，可以使用这个变量名进行函数调用，在上面的代码块中定义了add变量，让我们来调用他：

add(1, 2); // 返回：3



6. 返回值语句

return语句终止函数的执行，并返回一个指定的值给函数调用者。

Return语句可以在函数中使用，如下：

let add = fn(a, b) {

// return 语句

return a + b;

}



## 1.2 编译器：词法分析、语法分析

根据以上编程语言的设计，需要编写一个编译器进行支持。

该编译器需要支持：词法分析、语法分析。

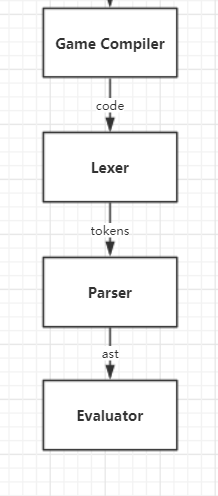


图1.1 编译流程图

### 1.2.1 词法分析

词法分析是字符序列转换为单词（Token）序列的过程。词法分析阶段是编译过程的第一个阶段，是编译的基础。这个阶段的任务是从左到右一个字符一个字符地读入程序代码，即对构成源程序的字符流进行扫描然后根据构词规则识别单词(也称单词符号或符号)。

根据1.1中所述，本程序的词法分析需要支持的token类型如下：

1. 常量： integer整形 string字符串 true/false 布尔类型；

2. 标识符：即变量名，本项目中定义为以连续字母组成，因为需要区分大小写的，所以字母可以是从“A”到“Z”的大写字母和从“a”到“z”的小写字母；

3. 界符：分号; 逗号, 左右小括号() 左右大括号{}；

4. 操作符：减号- 加号+ 乘号\* 除号/ 赋值= 全等比较== 不等比较！= 大于> 小于<；

5. 关键字、保留字：let if else null return fn while。

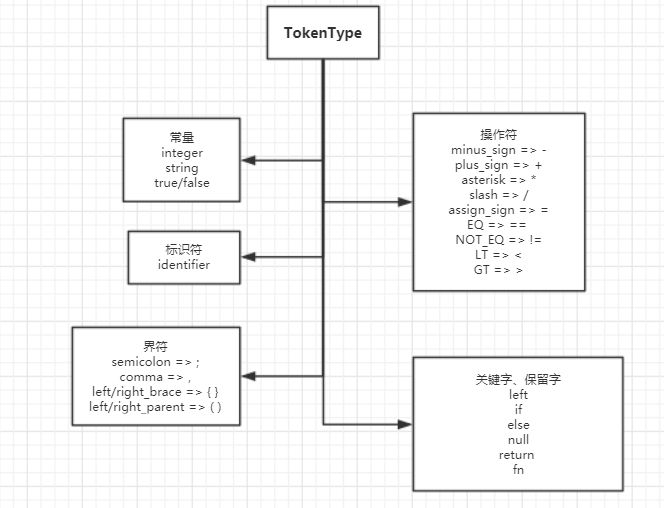
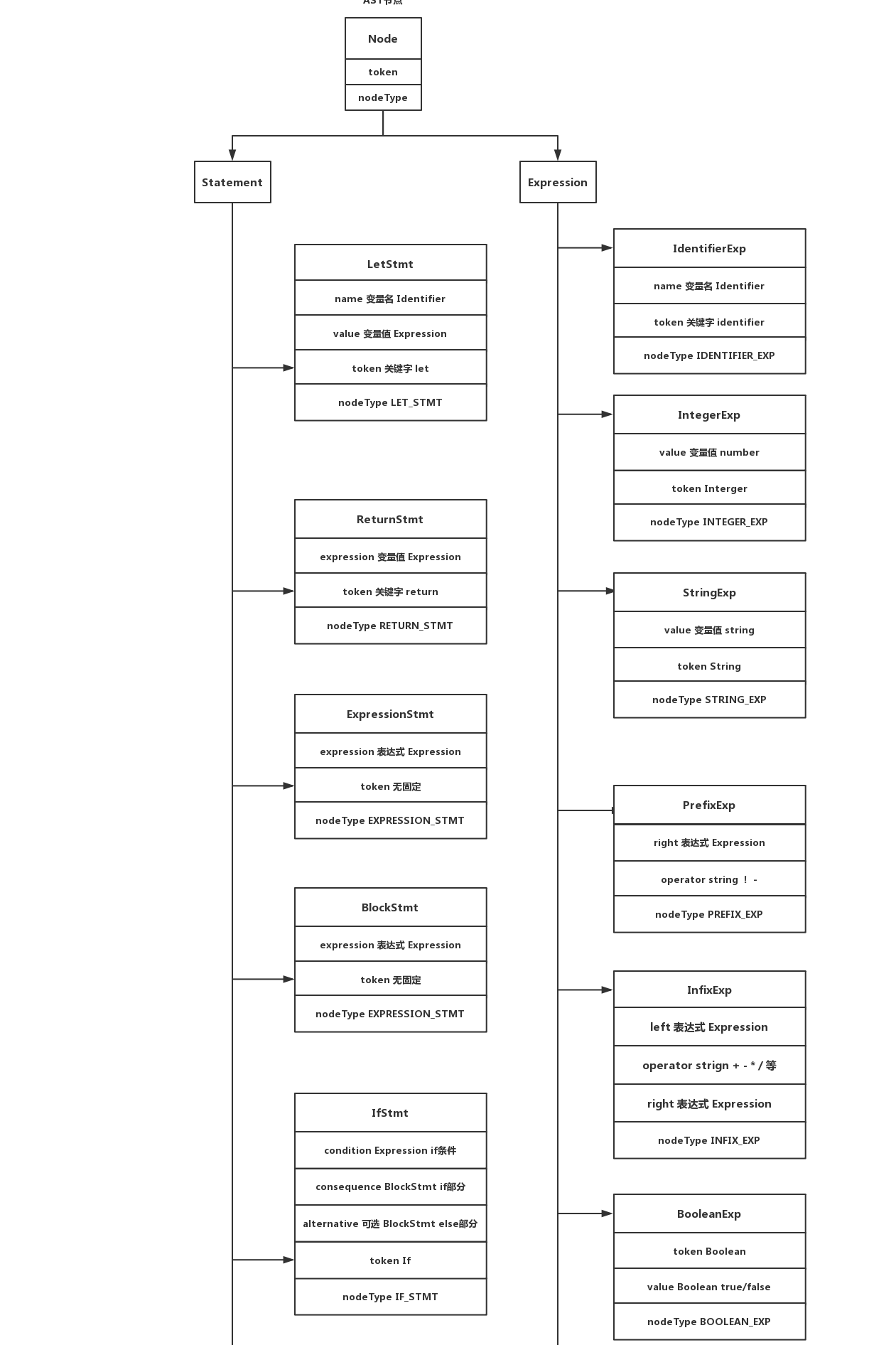


图1.2 词法分析支持的token类型图

### 1.2.2 语法分析

语法分析是编译过程的一个逻辑阶段。语法分析的任务是在词法分析的基础上将单词序列组合成各类语法短语，如“程序”，“语句”，“表达式”等等.语法分析程序判断源程序在结构上是否正确.



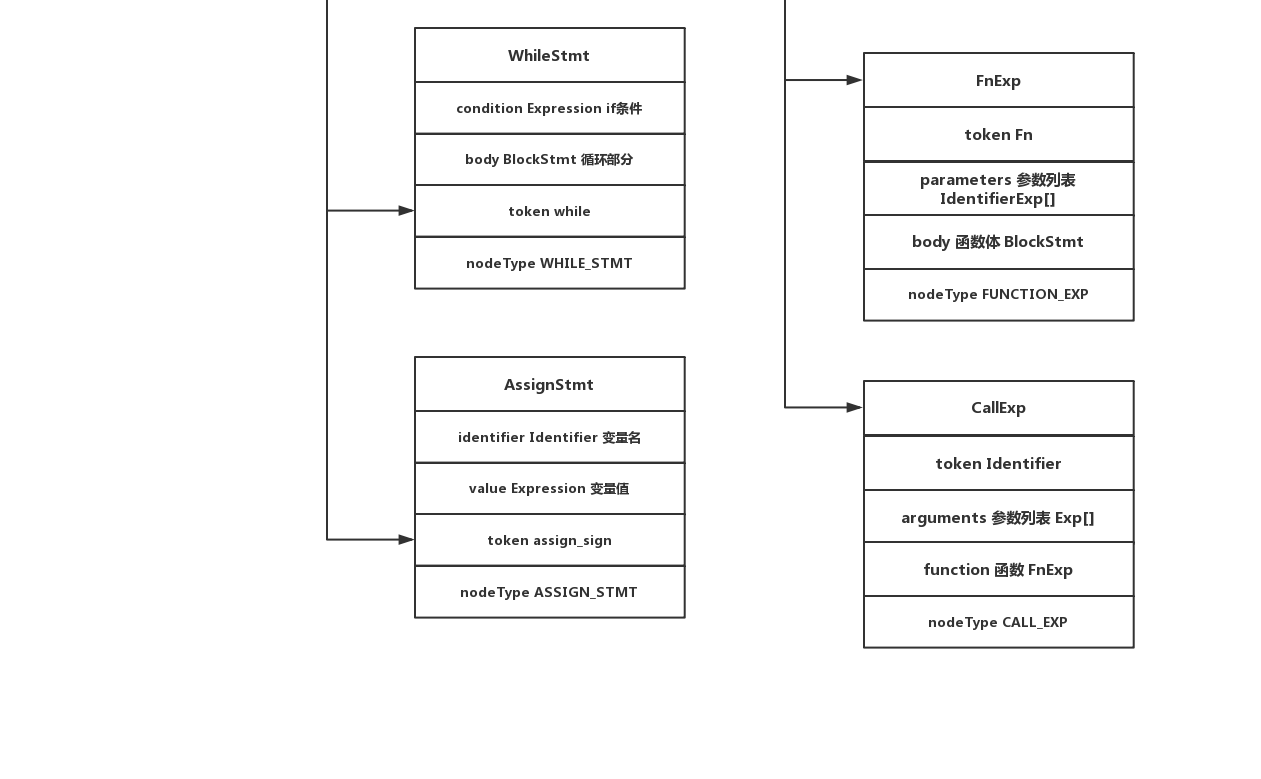


图1.3 语法分析支持的node类型图

根据1.1中所述，本程序的语法分析需要支持的语法如下：

**语句的解析：**

1. 声明语句的语法解析

2. 赋值语句的语法解析

3. 表达式语句的语法解析

4. 返回值语句的语法解析

5. 语句块的语法解析

6. 条件判断语句的语法解析

7. 循环语句的语法解析

**表达式的解析：**

1. 变量表达式的语法解析

2. 整形表达式的语法解析

3. 字符串表达式的语法解析

4. 布尔值表达式的语法解析

5. 函数表达式的语法解析

6. 前缀表达式的语法解析

7. 中缀表达式的语法解析

8. 函数调用表达式的语法解析

### 1.2.3 程序执行器

程序执行器是本“微型编译器”的最终阶段，他将接收语法解析后的结果，依托于javascript的架构进行代码的执行，并输出结果。该阶段可以验证源程序在语义上是否符合规范。

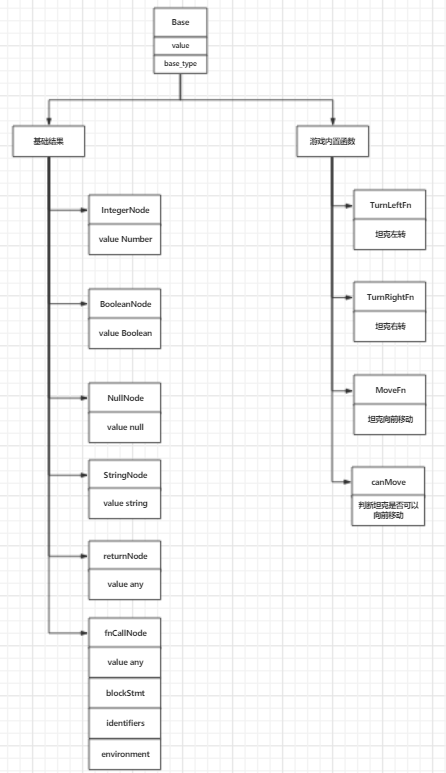


图1.4 程序执行器支持的类型

在语句的执行过程中，程序执行器将支持产生如下结果：

基础结果：

1. 基础类型节点：整型节点、布尔值节点、字符串节点、null（空）节点；

2. 返回值类型节点；

3. 函数表达式节点。

同时，为了支持编程游戏模式，需要支持内置函数的调用：

1. Move(step): 控制坦克向前移动；

2. turnLeft()、turnRight()：控制坦克向左、向右转向；

3. canMove(): 用于判断坦克是否可以向前移动。

## 1.3 编程平台设计：编程关卡

本编程平台关卡设计：通过编写代码控制坦克移动 转向 以到达 目的地。

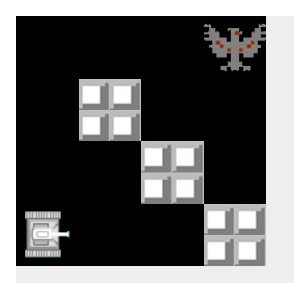


图1.5 编程平台关卡界面示意图

1. 左下角为用户控制的坦克

2. 纯黑色方格即为坦克可以行驶的“路”

3. 白色的钢铁即为路障，用于阻碍坦克行驶

4. 右上角的“鹰形”标志即为目的地，用户要控制坦克形式到这里。

基于通关模式的设计，编程游戏需要内置4个函数供用户调用：move(step)向前移动、turnLeft()向左转向、turnRight()向右转向、canMove()用于判断坦克是否可以向前移动。

在编程平台关卡设计上，结合上述编程语言特性，既要有简单的语法特性，又要保持复杂度中等的编程思想：循环、条件判断语句、函数调用等。基于此，需要设计特定的关卡形式引导用户进行相应的思考，并对用户编程思想层次进行分层：

1. 基础阶段：用户需要掌握基本的内置函数调用。

2. 循环、条件阶段：要求用户使用循环语句精简重复性的代码，使用条件判断进行情况的判断以执行不同的逻辑语句。

3. 函数声明、调用阶段：鼓励用户将具有一定特殊意义的代码段封装为一个可重复的调用的函数，利用函数调用来优化代码，提高代码可读性。

### 1.3.1 基础阶段

在此阶段，用户需要掌握基本的内置函数调用。



图1.6 编程平台关卡一界面

关卡解法：用户可以直接输入 move(5)

控制坦克向前移动5格到达目的地。

设计目的：引导用户调用最基本的move(step)函数，同时用户可以熟悉

1. 游戏的通关模式；

2. 函数的调用，参数的传递。

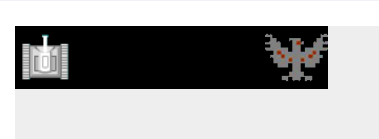


图1.7 编程平台关卡二界面

关卡解法：在关卡一的基础上，用户需要预先调用turnRight()函数进行转向

turnRight();

move(5);

设计目的：承接关卡一，用户需要接受坦克方向这个概念，并且调用正确的内置转向函数，调转坦克方向对准目的地，并使用move(step)内置移动函数控制坦克的移动。

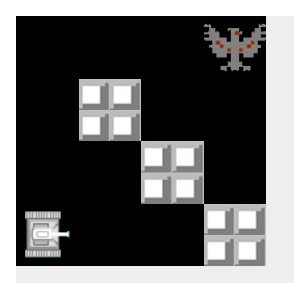


图1.8 编程平台关卡三界面

关卡解法：关卡三融合了关卡一、关卡二，用户首先需要调用turnLeft()内置左转函数，然后向前行驶，在行驶三格后，需要调用turnRight()内置右转函数，之后继续行驶，抵达目的地。

turnLeft();

move(3);

turnRight();

move(3);

在基础阶段，主要是引导用户熟悉游戏模式，了解游戏通关模式，并会调用简单的内置函数，对代码编程有一个整体的认识。

### 1.3.2 循环、条件判断阶段

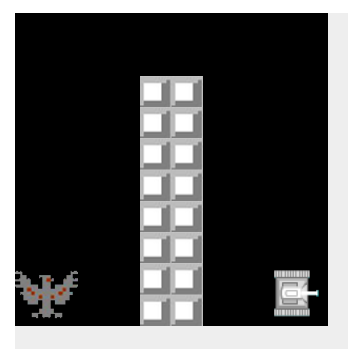


图1.9 编程平台关卡四界面

关卡解法：在通关了关卡三后，用户可能因为前面的通关经验写出如下代码：

turnLeft();

move(4);

turnLeft();

move(4);

turnLeft();

move(4);

可以看出，这段通关代码中将turnLeft() move(4); 重复了三遍，基于此，可以引导用户写出具有循环的代码：

let times = 3;

while(times) {

turnLeft();

move(4);

times = times - 1;

}

在本关卡中，用户可以学习到：

1. let声明语句语法

2. 变量值的概念：整形、布尔值

3. 对于整形数值的基本运算赋值

4. 布尔值的判断

5. 循环while语句的语法

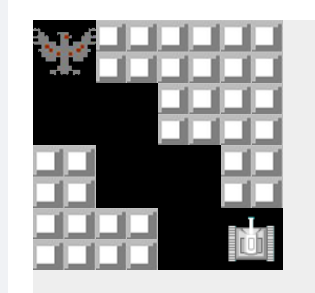


图1.9 编程平台关卡五界面

关卡解法：在关卡四的基础上，用户可以继续熟悉while循环的代码思想：

let times = 3;

while(times) {

turnLeft();

move(1);

turnRight();

move(1);

times = times - 1;

}



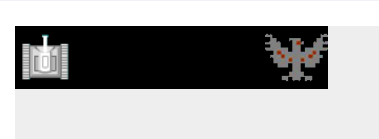


图1.10 编程平台关卡一、二界面

可以看出，关卡一、关卡二不同之处在于一开始的坦克方向，根据方向的不同，用户需要调用turn系列方法进行调整，在这里，其实是一种条件判断的思想。

关卡通用解法：

if(canMove()) {

turnRight();

}

move(5);

在本关卡中，用户可以学习到：

1. if语句的语法

2. canMove()函数调用的返回值机制

3. 布尔值的判断

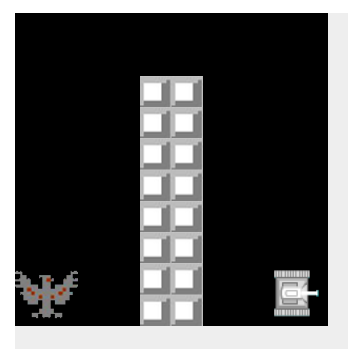


图1.11 编程平台关卡四界面

在循环的基础上，可以与条件判断进行组合使用：

let times = 6;

while(times) {

times = times - 1;

if (canMove()) {

move(5);

} else {

turnLeft();

}

}

在循环、条件阶段，要求用户使用循环语句精简重复性的代码，使用条件判断进行情况的判断以执行不同的逻辑语句。在此阶段，逐渐培养用户代码编程思想。

## 1.3.3 函数声明、调用阶段

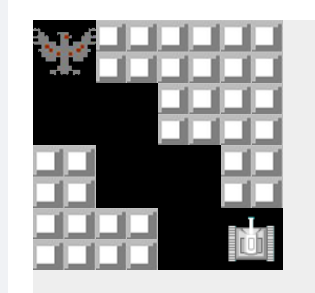


图1.12 编程平台关卡五界面

先前解法：

let times = 3;

while(times) {

turnLeft();

move(1);

turnRight();

move(1);

times = times - 1;

}

可以看出：向左转，前进一步，向右转，前进一步，这段代码在循环中其实代表着的是一个小循环步骤，具有一定的特殊意义（称为一个步骤），合乎编程思想的函数思想，所以，进一步的，这里其实可以抽象为一个step函数，用于调用，使得代码逻辑会更加清晰.

let step = fn() {

turnLeft();

move(1);

turnRight();

move(1);

}

let times = 3;

while(times > 0) {

step();

times = times - 1;

}

在函数声明、调用阶段，鼓励用户将具有一定特殊意义的代码段封装为一个可重复的调用的函数，利用函数调用来优化代码，提高代码可读性，同时有利于代码的可维护性。

## 1.5 本章小结

本章主要是对少儿编程平台系统的需求进行总结并分析，依次介绍了完成本系统所需要的编程语言、编译器所必须要支持的更方面功能、以及游戏关卡的思路设计。对编程语言的语法进行详细的声明定义，对编译器的词法分析的单位类型划分、对语法分析所需要支持的语言、对程序执行所需要支持的执行结果进行定义。