**JavaScript的C++解释器**

**系统设计说明文档**

负责人：张驰

项目成员：谭秦兵 郑天季 张乾坤

2016年1月15日

**目录**

[1 引言 3](#_Toc440049769)

[1.1 文档编制的目的 3](#_Toc440049770)

[1.2 背景 3](#_Toc440049771)

[1.3 其他 3](#_Toc440049772)

[2 设计概述 3](#_Toc440049773)

[2.1 任务和目标 3](#_Toc440049774)

[2.2 开发与运行环境 4](#_Toc440049775)

[2.2 项目开发工具 4](#_Toc440049776)

[2.3 条件与限制 4](#_Toc440049777)

[3 系统详细设计 4](#_Toc440049778)

[3.1 系统结构设计及子系统划分 4](#_Toc440049779)

[3.2 系统功能模块设计详细 5](#_Toc440049780)

[3.2.1 Global类 5](#_Toc440049781)

[3.2.2 Element类 5](#_Toc440049782)

[3.2.3 Function类 6](#_Toc440049783)

[3.2.4 Util类 7](#_Toc440049784)

[3.2.5 Parser类 8](#_Toc440049785)

[3.2.6 Calculator类 8](#_Toc440049786)

[3.3 main主程序控制](#_Toc440049788) 9

[4 项目难点及解决方案 11](#_Toc440049789)

[4.1 字符串处理 11](#_Toc440049790)

[4.2 跳转循环 13](#_Toc440049791)

[4.3 数据存储 14](#_Toc440049792)

[4.4 函数递归 14](#_Toc440049792)

[4.5 数组递归定义 14](#_Toc440049792)

[4.3 没有语法树纯手写 14](#_Toc440049792)

[5 程序使用方法 15](#_Toc440049793)

[6 致谢](#_Toc440049794) 11

# 1 引言

## 文档编制的目的

本文档旨在记录JavaScript的C++解释器在设计过程中所使用的技术以及克服的困难，为后期的代码批注、审阅、修改等工作提供一个初步的参考。

此外，本文档作为本次项目的一个记录，也同时能为其它相关项目提供技术支持，让后来者少走弯路。

## 背景

JavaScript的C++解释器这个软件专门为2015-2016秋冬学期浙江大学程序设计方法学课程所设计。通过使用C++以及程序设计语言方面的知识，实现了针对JavaScript程序设计语言的解释执行，凡是符合本系统要求的JavaScript语法，输入到本程序当中，都可以解释执行得到相应的结果。综合展示了C++的强大应用以及程序语言在解释过程中的具体过程。说明了项目组成员对JavaScript语言的理解程度，对C++语言熟练的使用能力以及对程序语言设计当中一些分析方法的深入理解。

## 其他

在本项目中，张驰作为项目负责人，负责系统构架设计、表达式计算方法设计与实现；谭秦兵负责语法中语句分析设计与实现，对JavaScript中的关键词进行分词；郑天季负责JavaScript语言main和输入输出接口设计、报告撰写、ppt与视频制作；张乾坤负责对系统进行测试。

我们的项目完全由C++标准库编写，没有其他依赖，并已在GitHub上开源，项目地址为

https://github.com/WellyZhang/ICJs。

# 设计概述

## 任务和目标

JavaScript的C++解释器作为一个展示课程所学及自身能力的项目，实现如下功能与效果：

* 支持JavaScript中var类型的多变量定义
* 支持JavaScript中表达式的计算和结果的返回
* 支持JavaScript中单变量的赋值和多变量的同时赋值
* 支持JavaScript中for-each循环
* 支持JavaScript中switch类型的循环运算
* 支持JavaScript中if和else的类型选择
* 支持JavaScript中while类型的循环运算
* 支持JavaScript中输出结果的显示
* 支持JavaScript中函数的定义与运算及多值返回
* 支持加减乘除与或非，判等，大于小于，大于等于，小于等于，括号等运算符号
* 支持字符串的链接，函数的嵌套递归运算，数组中的多类型变量
* 提供命令行方式使用及文件方式运行

## 2.2 开发与运行环境

本程序基于Visual Studio 2013版本的C++语言开发并依赖Windows.h编程库，理论上能够在任何合理配置C++标准库的Windows编程环境上运行。在实际测试中，我们发现，程序能正确地在Windows 7及Windows 10两个平台上运行。由于程序并不依赖Windows.h库文件，因此从理论上讲Linux各发行版本及Mac OS系统都可以使用本程序。

## 项目开发工具

本程序在系统构架上完全依赖于C++标准库，并未使用任何其他的程序设计库。

## 条件与限制

本项目以MIT开源协议开源，项目使用者可以免费地对程序进行任意的修改、删除、分发等操作，而无需与本项目组达成商业协议。

# 系统详细设计

## 系统结构设计及子系统划分

本程序总共被分为七大部分，它们分别是：

* Global类——记录程序中所用到的全局变量
* Function类——负责记录JavaScript中遇到的函数变量中的参数
* Element类——负责记录JavaScript中遇到的每一个变量的值，类型，变量名
* Util类——程序设计当中的工具类，实现程序设计当中需要但是标准库里面并没有的功能
* Parser类——对程序中输入的语句进行此法分析，寻找并且处理程序里面的关键词
* Calculator类——用于程序中遇到的所有表达式的计算，包括使用堆栈进行符号数据的压栈出栈运算操作，对函数、数组的处理等
* main文件——负责整个程序的输入输出，是一个面向用户的接口，用户通过main文件输入自己想要运行的程序代码并且通过main得到相应的输出

其中main文件是程序的主文件，用于程序的整体控制。其余各类相互平行，没有继承关系。

## 系统功能模块设计详细

### globals文件

globals文件包含十个全局变量：

* static const int \_undefined = 0;
* static const int \_null = 1;
* static const int \_string = 2;
* static const int \_number = 3;
* static const int \_boolean = 4;
* static const int \_array = 5;
* static const int \_object = 6;
* static const int \_function = 7;

上面是对几个变量的类型值进行定义，从string到null到数组以及函数的变量类型都用相应的值方便的表示出来。

此外，还有

* static const int \_ok = 0;
* static const int \_fault = -1;

对是与否两个类型的值进行定义，表征运算过程中是否出现错误。

### Element类

Element类负责对程序中变量进行处理。

我们整个解释器一共实现了五种类型的数据类型，分别是string，double，bool，数组和函数。由于var是多变量类型，在Element类初始化的时候要将data的类型根据用户实际传入类型进行空间申请，所以在Element的构造函数中做了这样的一个操作：

class Element

{

public:

std::string key;

int type;

any\_t data;

Element(){ key = ""; type = Global::\_undefined; data = NULL; };

Element(std::string \_key, int \_type, any\_t \_data=NULL){

key = \_key;

type = \_type;

switch (type){

case 0:case 1:data = NULL;

break;

case 2:data = (std::string\*)\_data;

break;

case 3:data = (double\*)\_data;

break;

case 4:data = (bool\*)\_data;

break;

case 5:data = (std::vector<Element>\*)\_data;

break;

case 6:break;

case 7:data = (Function\*)\_data;

break;

default:data = NULL;

}

}

}

### Function类

函数是程序中一个较为特殊的变量类型，所以在程序设计当中我们给Function这样的一个特殊的类型专门设计一个类来进行描述，在Function类里面，我们使用函数名字、函数参数类型名、函数域内主体来进行描述。其中函数名的类型是单独的string类型，参数存储成一个string的vector类型，函数主体也存储成一个单独的string的vector类型用来表示函数内部的内容。具体实现如下：

class Function

{

public:

std::string key;

std::vector<std::string> param\_names;

std::vector<std::string> body;

};

### Util类

Util类是专门为compute计算文件设计的一个工具类，包括对语句进行分割，统计字符出现个数，数组字符串转换函数，具体内容如下：

static void split(std::string &s, std::string delim, std::vector<std::string> \*ret, bool preserveBlank);

//进行字符串的分割

static std::string &trim(std::string &s);

//删除字符串中的空格

static int numOfChar(std::string &s, char c);

//统计字符串中字符c的数量

static std::string arrayToString(std::vector<Element> ary);

//将一个数组返回成字符串类型

### Parser类

Parser类主要负责的是词法分析。文件里面包含一些分析过程中用到的工具函数，并且提供给main主函数相应的函数接口，接口函数为parse，在接口当中，Parser类传入从main输入得到的程序数据，然后再直接在\_parse函数中进行处理。因为语言需要由分号进行对程序的区分，所以在最开始的时候需要对分号进行一下处理，以保证语句的完整性。之后的整体逻辑就是if和else对各个关键词进行处理，针对每一个关键词在内部交给各个逻辑，例如遇见var的时候，查看var后面跟着几个变量，还需要判断是否对这个变量进行初始化，初始化的时候判断数值的类型，如果初始化的时候后面跟着的是表达式，那么还要将表达式交给calculator进行运算，然后将得到的值初始化给前面的变量。如果是if关键字的话需要首先判断两个小括号，之后取得括号里面的内容交给calculator进行处理，处理以后根据相应的判断来进行下一步的操作。一次类推，在Parser类里面主要对下列关键字进行了详细的处理：

if (oper == "var"){

else if (oper == "if"){

else if (oper == "else"){

else if (oper == "for"){

else if (oper == "switch"){

else if (oper == "case"){

else if (oper == "default"){

else if (oper == "break"){

else if (oper == "continue"){

else if (oper == "while"){

else if (oper == "return"){

else if (oper == "function"){

else if (oper == "{"||oper=="}")

if (it->find("=") != string::npos && it->find("==") != it->find("="))

更详细的处理过程见源代码中ICJs\_parser.cpp文件里面的具体操作

在Parser类里面，除了上面的一些主要的操作，还包含了下面的几个工具函数

void clear\_string(string& str)

//将字符串里面的内容清空掉

void string\_dblank(string& str)

删除字符串里面所有的空格符号

vsit vector\_find\_bracket(vector<string> &lines, vsit line)

//查找大括号

vsit vector\_find\_else(vector<string> &lines, vsit line)

//查找else关键词

vsit vector\_find\_next\_case(vector<string> &lines, vsit line)

//查找下一个case关键词的位置

int forloop(vsit &bg, vsit&ed,

map<std::string, Element> &variables,

vector<Element> &output,

vector<Element> &fun\_ret,

string var,

string list)

//for循环操作

void split(string& s, string delim, vector<string> &ret)

//拆分关键词的操作

int Parser::run\_func(Function &func,

std::map<std::string, Element> &variables,

std::vector<Element> parameters,

std::vector<Element> &ret,

std::vector<Element> &output)

//运行函数的操作

### Calculator类

Calculator类主要是进行表达式计算的类，用来进行程序运行当中一些必要表达式，数组，函数的加减乘除与或非大小与比较相等判断等等操作

调用方法：

SolarSystem \*galaxy = new SolarSystem();

库依赖：

* Windows.h
* ICJs\_compute.h
* ICJs\_util.h
* ICJs\_types.h
* ICJs\_parser.h
* stack
* cmath
* ctype.h
* sstream
* iostream

类成员：

* public:
  + static int calculate(std::string &exp, std::map<std::string, Element> &variables, std::vector<Element> &rets,std::vector<Element> &output);

功能：对数组元素进行计算

* + static int numeric(std::string &exp,std::map<std::string, Element> &variables,std::vector<Element> &rets,std::vector<Element> &output);

功能：对数值，逻辑以及字符串表达式进行计算

* + static int isOperator(std::string input);

功能：判断是否是我们程序所支持的运算符

* + static int isFunction(std::string input, std::map<std::string, Element> &variables);

功能：判断数据类型是否是函数类型

* + static int isNumVar(std::string input, std::map<std::string, Element> &variables);

功能：判断该var类型是否是数字类型

* + static int RPNCalc(std::string input, std::map<std::string, Element> &variables, Element &ret);

功能：利用堆栈数据结构进行表达式的计算

* + static int priority(std::string opt);

功能：设置运算的优先级

* + static int isStringVar(std::string input, std::map<std::string, Element> &variables);

功能：判断该var类型是否是字符串类型

* + static int isLogic(std::vector<std::string> inputs);

功能：判断该数据类型是否是逻辑类型

* + static int isComma(std::string input);

功能：判断是否是我们程序支持的括号

* + static int isLogicOperator(std::string input);

功能：判断是否是我们程序所支持的逻辑操作

* + static int isArrayVar(std::string input, std::map<std::string, Element> &variables);

功能： 判断该数据类型是否是数组类型

## main主程序控制

main主程序按照如下逻辑控制项目的运行：

1. 程序开始运行时，首先初始化系统，显示用户提示模块以及命令行的第一行，给用户输入提示以及程序声明。
2. 用户输入命令，程序进行命令判断，如果输入的是exit退出命令直接退出程序，如果是%load则使用文件加载处理方式，让用户指定文件中的内容成为本程序的输入内容。否则，用户手动输入内容并且存储到缓存区，使用parse接口对用户输入数据进行处理。
3. 在文件处理过后找到用户想要输出变量的内容，从parse接口返回的内容中获取所需要的数据输出。如果并没有返回要求的输出，则返回空；如果程序出错，则返回“Error ”。

# 项目难点及解决方案

## 字符串处理

字符串处理是本程序中最为核心的一个部分，在如何处理用户输入的较为繁杂的数据过程当中我们遇到了相当多的问题，包括如何进行分词，进行关键词的提取，进行表达式的拆分计算。在解决这个问题的过程当中，我们首先想到的是使用空格字符进行此法的拆分，之后根据由空格拆分而成的字符段进行关键词的分析处理，具体逻辑包括对中括号，大括号的提取，对“=”号的提取，这些特殊运算符在我们的词法分析中起到了至关重要的作用

## 循环跳转

为每个循环跳转语句建立新的语句段、定义域，递归调用相应API执行，continue、break直接返回语句段的段首或段尾即可，控制变量的检查则在调用的上一个层次进行，这样也可以方便地添加和销毁本地变量。

## 函数递归

函数作为我们解释器的一个拓展项目，也是我们遇到的较复杂的问题之一，其主要困扰我们的是如何进行形如f1（f2（）+1）这样的函数相互嵌套表达式处理

## 数组递归定义

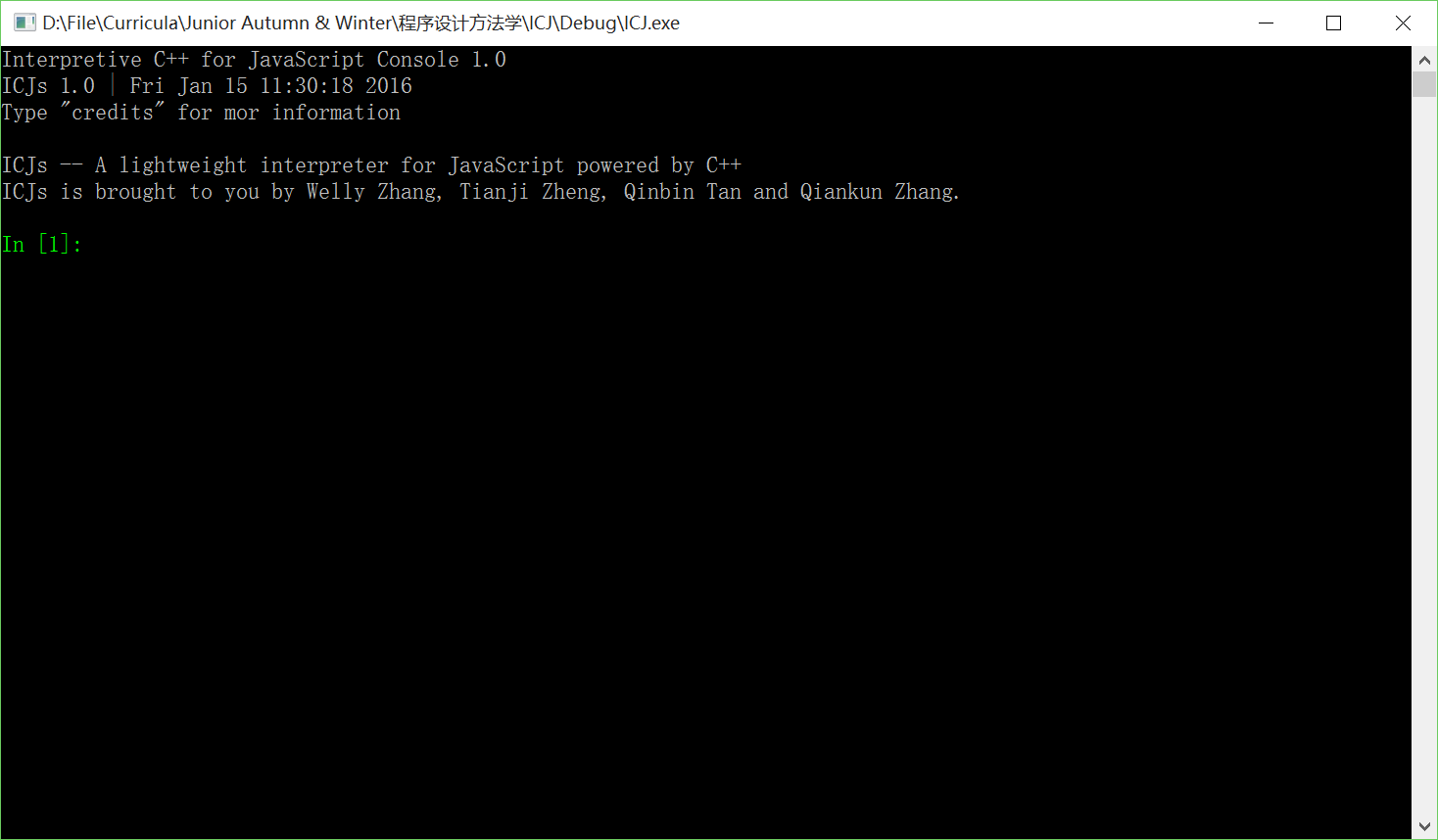
同函数递归的逻辑相同，数组是我们程序设计当中的一个拓展项目，同样考虑到形如a1[a2[]+1]的表达式，如何对这样的表达式进行计算就成了一个较大的问题，当然，我们首先想到的就是利用递归去逐层次来求得数组的值并且返回，

## 在没有语法树的情况下手写

主要难点在于每种语句都要单独处理字符问题，提取语句的不同部分，因此我们对一些语句的格式进行了一定的限定，例如if（）之后必须换行等等，减少了一部分代码复杂程度，通过对字符串库的灵活运用，实现分句分词，并执行的功能。

# 程序使用方法

运行该程序后，用户将会首先看到用户欢迎界面。之后，用户即可根据JavaScript语法输入对应语句，根据Out输出辨识程序的返回值。此外，用户可以输入“credits”阅读项目的致谢内容。



图一 ICJs项目欢迎界面

# 致谢

在程序设计过程中，我们得到了诸多人员的无私帮助。我们尤其感谢翁恺老师对项目成员的亲切指导；感谢室友们对项目组成员工作的理解与支持。此外，我们还要感谢诸多无私的网友，他们或者提供了部分技术支持，或者撰写博客解释了C++和JavaScript一些特殊的应用，或者属于专业技术开发人员，或者在Google Group中与大家热烈地进行了讨论。

我们从这些交流中受益匪浅。没有他们的帮助，本项目必然不可能在规定的时间内完成。为此，我们在此对他们的帮助表示衷心的感谢！