1. 介绍

《s • C • ratch 高级开发框架》是一款由不同之者设计的，基于 C++和 scratch 编程语言实现的，用于提高 scratch 创作效率的开发框架。

该框架包含两个组件，一个是基于 scratch 实现的 FSC Plus，另一个是基于 C++实现的 C 语言编译器。

该框架的使用流程是：使用 C 语言编写程序→用该 C 语言编译器将程序编译成 FSC Plus 类汇编语言→在 scratch 中编写 C 语言中调用了的 scratch 函数→运行 FSC。

该框架使用的 C 语言并非传统的 C 语言，而是有着一套新的标准。请在使用前详细阅读标准。

本项目为开源项目，如需使用可自行操作。

本文档主要叙述该 C 语言的标准，FSC Plus 的标准见附件。

1.1 快速入门

完整介绍该项目需要较大篇幅，接下来将阐述一些概要以便快速入门。

1.1.1 从 C 语言到 s • C • ratch

快速入门 s • C • ratch 要求用户有 C 语言基础。只需注意几个与 C 语言有明显区别的地方。

C 语言有很多变量类型，但是 s • C • ratch 只有 **var** 类型，即万能变量。

比如，在 C 语言中，定义变量是这样的

```
int a;
```

但是在 s•C•ratch 中，你需要这样

```
var a;
```

在 C 语言中，一般使用 **printf** 来输出，而且需要引入头文件；而在 s•C•ratch 中，不能也不需要引入头文件，输出可以使用 **display** 函数，只有一个参数。输出到 scratch 的 **console** 列表中。

1.1.2 C 语言与 scratch 的交互

如果光是运行 C 语言，那这个项目便失去了意义。s•C•ratch 的核心功能是 C 语言与 scratch 的交互。

若想使用 C 语言调用 scratch 的函数，请在 C 语言中使用函数 **scr_func**。该函数有两个参数，都是 **var** 类型，第一个是 **id**，第二个是 **arg**。还有一个 **var** 类型的返回值。**id** 表示你要调用的函数在 scratch 中的编号（编号是你自己决定的），**arg** 表示向 scratch 的函数传递的参数。参数只能传递一个，但是你可以传递指针，指向数组或结构体，间接实现传多个参数。

在 scratch 中，你需要在函数 **call_scratch_function** 中增加分支，分支的条件是 **id** 的值。在每个分支中编写函数的内容，建议单开函数而不是在 **call_scratch_function** 中直接写。若需要返回一个值，请使用 **function_return** 函数。

这里的流程较为抽象，不过该项目提供了一个具体案例：利用 C 语言调用 scratch 的 **scf_test** 函数，以将 scratch 变量 **scf_test** 的值更改为 1919810，该函数会返回 114514，并利用 C 语言将该返回值打印出来。该案例 C 语言代码如下

```
var main(){var tmp=scr_func(1,1919810);display(tmp);}
```

1.1.3 编写你的 C 语言代码

开发环境建议选择 Microsoft Visual Studio。

将文件 sCr_Template.zip 复制到

Documents\Visual Studio 2022\Templates\ProjectTemplates

文件夹下即可将模板导入 Visual Studio。

之后使用该模板创建项目并编写 C 语言代码。

1.1.4 编译你的 C 语言代码

写好 C 语言代码后，先粘贴到**代码转换器**，该程序可以将您的代码去除注释并压缩成一行。然后粘贴到 **C 语言编译器**进行编译，程序会输出若干行类汇编代码。这就是你的代码的编译结果。

1.1.5 运行你的代码

将编译结果复制，粘贴到 scratch 程序 **FSC Plus** 中的列表 **original_code** 中，运行 scratch 程序即可。

1.1.5 Q&A

Q：提示“错误：列 0 你的代码有语法结构错误，但是做这个编译器的人太菜，不知道你哪错了。你自己把代码拿到别的编译器测测吧。”

A：由于多方面因素，该编译器不能准确报出所有错误，因此你需要利用其他编译器自行检查。（很大可能是你写了 **int** 而不是 **var**）

Q：写了 **main** 函数，但是仍然提示“错误：列 0 没有找到 main 函数。”

A：这个错误很多情况下都会报，很可能是你的代码结构错误，比如少了大括号、少了分号等等。

Q：运行结果与理论上不一致。

A：该项目尚处测试阶段，存在大量潜在 bug，在开发 scratch 作品时请勿过度依赖此工具。

2. 编译过程

编译大体分为以下三个过程：词法分析，语法分析，语义分析及目标代码生成。

2.1 词法分析

通过词法分析得到 token 序列。每个 token 有两个属性：type 和 content。type 表示该 token 的类型，分为 KEYWORD、IDENTIFIER、CONSTANT、OPERATOR 和 SEPARATOR。content 表示 token 的内容，其值为代码本身。

2.2 语法分析

语法分析器基于 ebnf 进行枚举生成抽象语法树。

该 C 语言的 ebnf 见附件。

该语法分析非预测分析，因此无法报出语法结构错误具体位置。

2.3 语义分析及目标代码生成

该编译器不生成中间代码，在语义分析过程中直接生成目标代码。

目标代码为 FSC 代码，需使用 FSC Plus 运行。

该语义分析器不具有纠错功能，因此一次最多报出一处错误。

3. 系统

受 scratch 特性影响，该 C 语言与传统 C 语言的系统有很大差异。

3.1 数据

数据类型只有 var 一种，代表 scratch 变量的数据类型。

3.2 内存

每个整数对应一个内存地址，每个地址存储一个 var 类型变量。

3.2.1 内存空间

受 FSC Plus 特性影响，存储空间大小固定为 200000，地址从 0 到 199999。

地址 [1,99999) 为静态存储区，存放全局变量、全局数组、字符串常量。

地址 [100005,199999）为栈区，存放函数参数、局部变量、函数返回值。

3.2.2 内存空间分配

该 C 语言编译器不使用传统 C 语言的对齐式分配。所有数据会紧凑地按地址从小到大填满空间。

函数在栈上的空间分配按地址从小到大分别为参数，返回值，局部变量，控制链和机器状态。

3.2.3 特殊空间分配

[0] 为常量 0，不可修改。

[99999,100004]，[199999] 为系统关键区域，修改可能会引发程序崩溃。

3.2.4 越界访问

访问 [1,200000) 以外的内存空间为未定义行为。

3.3 运算

若该运算符在 scratch 中存在，则直接使用 scratch 的运算符模块运算。

部分运算符在 scratch 中不存在：

次幂运算：$a^x = e^{x \ln a}$

逻辑布尔运算：使用 > 运算符将 **var** 类型转换为 **bool** 类型

二进制位运算：不支持

3.4 优化

该 C 语言编译器不对代码进行任何效率优化。

4. 语法规则

该 C 语言与传统 C 语言语法大体类似，但存在一些差别，主要是为了适应 scratch 环境而做出的调整。

4.1 词法

4.1.1 KEYWORD

KEYWORD 为关键字类型。

关键字包括：**var**、**struct**、**void**、**if**、**else**、**switch**、**case**、**default**、**break**、**for**、**while**、**do**、**continue**、**const**、**return**。

4.1.2 IDENTIFIER

IDENTIFIER 为标识符类型。

标识符必须仅由数字、字母和下划线组成，并且必须以字母或下划线开头。

标识符不能是关键字。

4.1.3 CONSTANT

CONSTANT 为常量类型。

常量包括：数字、字符、字符串。

数字前加 0x、0b 分别表示十六进制、二进制。

在两个单引号之间只允许有一个字符或转义字符。

4.1.4 OPERATOR

OPERATOR 为运算符类型。

运算符包括：**->**、**++**、**--**、**<=**、**>=**、**==**、**!=**、**&&**、||、**+=**、**-=**、***=**、**/=**、**%=**、**(**、**)**、**[**、**]**、**.**、**!**、**+**、**-**、*****、**/**、**%**、**<**、**>**、**^**、**?**、**:**、**=**、**,**。

4.1.5 SEPARATOR

SEPARATOR 为分界符。

分界符包括：**<空格>**、**{**、**}**、**;**。

注意：分界符不包括换行，代码中不应当出现换行。

4.2 语法

具体语法规则请参考该项目的 ebnf 文件。

5. 语义规则

语义规则较为复杂，但大部分与 C 语言一致，故此处只介绍与 C 语言有差别的规则。

5.1 类型转换

任何类型的数据只要大小一致就可以相互转换，转换方式为内存拷贝。

5.2 数组

不支持数组初始化。

5.3 函数

函数参数数量必须固定。

函数参数可以传递指针，但删除了传递数组的写法。

5.4 结构体

不支持结构体初始化。

不支持在定义结构体的同时定义该结构体类型的变量。

不支持匿名结构体。

不支持在结构体中定义数组类型的成员。

5.5 表达式与运算

5.5.1 赋值运算

不考虑数据类型，只考虑数据大小。

5.5.2 次幂运算

^ 运算符不再表示按位异或，而是表示次幂运算，其优先级高于乘法运算且低于前置单目运算符。

5.6 流程控制

删除了 **switch** 语句。

6. 其他

该项目本身并不带有库。

6.1 内部实现函数

该项目存在内部实现的函数。

6.1.1 display

一个 **var** 类型参数 **arg**，没有返回值。

在 scratch 的列表 **console** 中打印数字。

每调用一次该函数会自动在列表中换行一次。

6.1.2 scr_func

两个 **var** 类型参数 **id** 和 **arg**，一个 **var** 类型返回值。

在 scratch 中调用编号为 **id** 的函数，并向其传递一个参数 **arg**。返回值由 scratch 代码决定。