Mao语言项目文档

MaoLang

周以恒

2015.12.31

Mao语言项目文档

一、编程环境

IDE: Clion1.2

建构工具: cmake

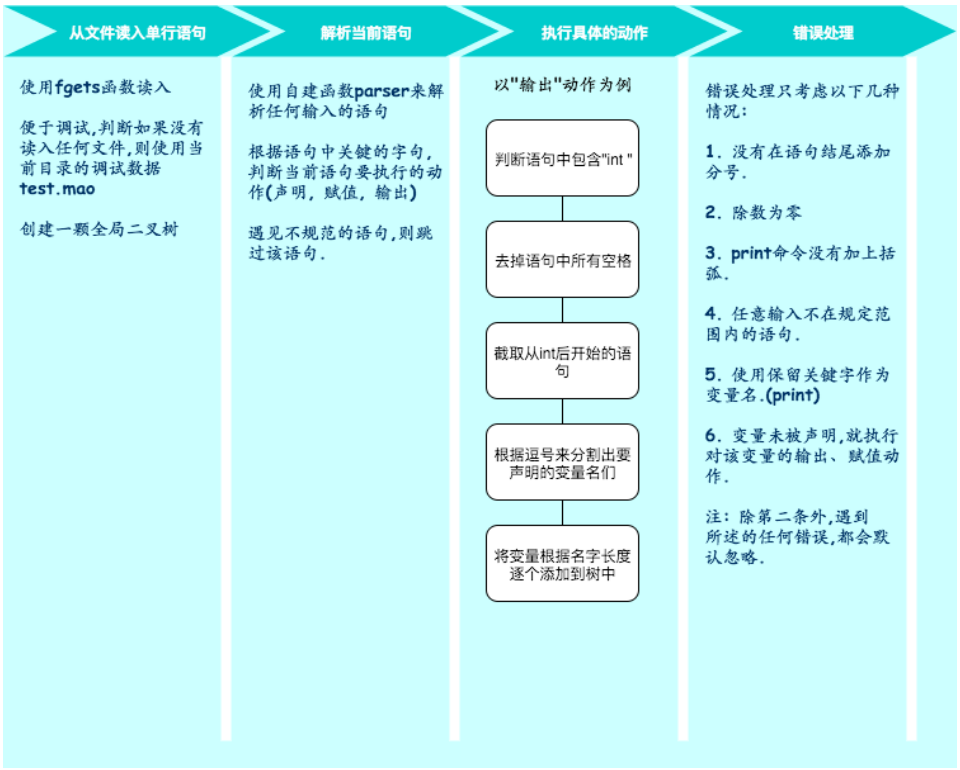
编译器: clang

标准: c11

编译: brew install cmake; cmake CMakeLists.txt; make;

运行: maolang “文件名”

二、简单流程图



三、二叉树的实现

1.Tree结构体中成员

- Node结构体类型的指针node，用于存储树枝中所包含的信息(变量名,数据)
- 自身Tree结构体类型的指针lchild，用于指向左树(节点)
- 自身Tree结构体类型的指针rchild，用于指向右树(节点)

```
typedef struct Tree {
    struct Node *node;  //An point structure including some detailed data
    struct Tree *lchild; // left children tree
    struct Tree *rchild; // right children tree
} *Tree;
```

2.Node结构体中成员

- string类型的数据varName，用于存储变量名(最大支持20位)
- union类型共用体包含了int和double两种数据类型，存储变量
- DataType类型包含了四种数据类型(double, int, operator, alpha), 用以判断变量的数据类型

```
/* It's a structure to store data of double and int types. */
typedef struct Node {
    char varName[21];
    union {             // Anonymous union
        int iData;
        double dData;
    } data;
    DataType dataType;  // type of data (Declared in utils.h)
} *Node;
```

3.二叉树的基本方法

- 创建并返回二叉树

```
/* Create a new binary tree and return it */
Tree createTree();
```

- 插入节点，数据都先以double类型传入，在内部根据DataType进行强制类型转换

```
/* Insert a variable with its name, type of data, and data into the tree. */
void insertVar(Tree, char *varName, DataType, double data);
```

- 通过变量名寻找节点

```
/* Find a node according to its name in the tree. */
Node findNode(Tree, char *);
```

- 已知一个节点，传入数据以改变该节点当前的数据

```
/* Find a node according to its name in the tree. */
Node findNode(Tree, char *);
```

四、赋值运算的实现

1.链表的建立

链表结构，内部存在element保存节点包含的数据，next指针指向下一节点

```
typedef struct Stack{
    struct StackEle StackELe;
    struct Stack *next;
} *Stack;
```

元素结构，其中DataType储存该元素的类型，用于以后运算和输出。

```c
typedef struct StackEle {
    union {
        double dv;
        int iv;
        char op;
        char al[101];
    };
    DataType type;
} StackEle;
```

2.计算表达式

将加减乘除，等于号，括弧都作为操作符，并根据它们在栈内外的位置来区分优先级。

```c
/* Get the priority of the operator which is out of the operator's stack. */
int getOsp(char operator){
    switch (operator){
        case '#':
            return 0;
        case '(':
            return 8;
        case ')':
            return 1;
        case '*':
            return 5;
        case '/':
            return 5;
        case '+':
            return 3;
        case '-':
            return 3;
        case '=':
            return 7;
    }
}
```

```c
/* Get the priority of the operator which is in the operator's stack. */
int getIsp(char operator){
    switch (operator){
        case '#':
            return 0;
        case '(':
            return 1;
        case '*':
            return 6;
        case '/':
            return 6;
        case '+':
            return 4;
        case '-':
            return 4;
        case '=':
            return 2;
    }
}
```

2.计算规则:

先建立三个栈，stack_number存常量数值(包括读取变量获得的数值),stack_operator存操作符,stack_alpha存将被赋值的变量.

(Isp: 栈顶操作符的优先级 ， Osp: 当前读取操作符的优先级)

当Isp小于Osp时: 将当前操作符写入stack_operator栈顶

当Isp大于Osp时(操作完毕后依然停留在当前操作符): (1)如果该操作符为'='，则将stack_number栈顶的数值赋给stack_alpha栈顶的变量;(2)如果是其他操作符,则计算连续两次取出stack_number栈顶数值,进行计算,然后写入stack_number栈顶.

当Isp等于Osp时(既判断左右括弧相遇时的情况): 将stack_operator栈顶操作符取出.

3.计算结果

在以上提到的运算过程中,一切函数之间参数的传递都是以stackEle为基础, 只有在具体涉及到加减乘除赋值的时候,才会取出element的具体数值.

计算结果最终会被返回,但是目前没有使用这一计算结果,因为, 在计算的过程中就已经为变量进行了赋值操作.

五、输出结果

1.print中可以包含的类型

变量和常量.

2.能否去掉括弧

不能,但是如果去除不会导致程序出错.

六、学习到的经验

1.想要实现泛型传递参数的时候可以使用可变参数

引入头文件<stdarg.h>

在参数位置使用”…”

```
Node updateNode(Node node, ...){
    va_list ap;
    va_start(ap, node);
    if(node->dataType == DOUBLE){
        node->data.dData = va_arg(ap, double);
    }else if(node->dataType == INT){
        node->data.iData = va_arg(ap, int);
    }
    va_end(ap);
    return node;
}
```

2.计算的过程中全部采用element来进行传递，只有在具体数值运算的时候，再去判断两个(或单个)element的具体数值.这样可以大大减轻代码冗余,使逻辑变得清晰.

3.二叉树并非当初想象的那样难, 只要清晰地把握节点遍历的反向与指针的变更即可.

4.以前在写作业的时候都是只有一个main.c, 开始写大项目以后, 认识到模块化组件化的重要性, 争取代码的复用, 减少冗余.