测试文档

测试数据一

```
{ Sample program
  in TINY language -
    computes factorial
}

read sum: { input an integer }

sum := 0;

for i:=0 to 3 do
    for j:=1 to 4 do
        sum += i*j;
    enddo;

enddo;

if (sum <> 10)
    write 1;

else

write 0;
```

```
输出
    "Program": [
    {
    "read": "sum"
     },
{
    "assign": {
      "assignExpression": [
        {
    "const": 0
}
        ;
],
"identifier": "sum",
."':="
      },
     for": {
          "body": [
          "bod,
{
   "for": {
    "body": [
                {
    "assign": {
        ianExpr
                      "assignExpression": [
                       "op": {
                           "leftExpression": [
                            {
    "identifier": "i"
}
                             "rightExpression": [
```

```
Vite App
     输入
                                                                        输出
                                                                                  "ElseBody": [
                                                                                  {
   "write": [
       { Sample program
        in TINY language -
        computes factorial
                                                                                       "const": 0
       read sum; { input an integer }
                                                                                    ]
                                                                                   }
       sum := 0;
                                                                                  "MainBody": [
       for i:=0 to 3 do
             for j:=1 to 4 do
                                                                                     "write": [
                sum += i*j;
                                                                                       "const": 1
              enddo;
       enddo;
       if (sum \Leftrightarrow 10)
             write 1;
                                                                                  "TestExpression": [
                                                                                  else
              write 0;
                                                                                      "leftExpression": [
                                                                                         "identifier": "sum"
                                                                                       }
                                                                                       ],
                                                                                       "rightExpression": [
                                                                                          "const": 10
                                                                                      }
                                                                                       "value": "<>"
```

测试数据二

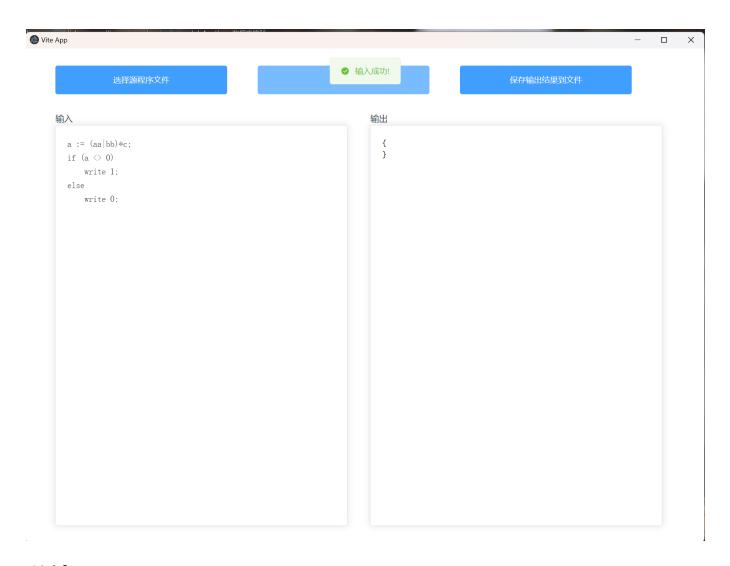
```
read x;
read y;
for i := x downto 1 do
    s := x \% y;
    s += 1;
    s := s \text{ and } 3;
    for j := 1 to y do
        m := 1 + (2*x);
        if (m<>0)
            write m;
        else
            write 0;
    repeat
     fact := fact * x;
     x := x - 1;
    until x = 0;
    enddo;
enddo;
if (not (x + y)*10 >= 100)
```

```
write 100;
else
write 0;
```

```
Vite App
                                                                                                                             □ ×
                                                              ❷ 启动成功!
    输入
                                                                    输出
       read x;
                                                                       "Program": [
      read y;
                                                                        {
"read": "x"
      for i := x downto 1 do
        s := x % y;
                                                                        },
         s += 1;
                                                                        {
                                                                         "read": "y"
         s := s and 3;
                                                                         },
         for j := 1 to y do
                                                                         {
           m := 1 + (2*_X);
                                                                          "for": {
                                                                         "for . .
"body": [
            if (m<>0)
                                                                          {
    "assign": {
              write m;
            else
                                                                                 "assignExpression": [
             write 0;
                                                                                 {
"op": {
         repeat
         fact := fact * x;
                                                                                    "leftExpression": [
           x := x - 1;
                                                                                     {
    "identifier": "x"
}
         until x = 0;
         enddo;
                                                                                      ],
       enddo;
                                                                                      "rightExpression": [
                                                                                     {
    "identifier": "y"
}
       if (not (x + y)*10 >= 100)
         write 100;
                                                                                     ],
                                                                                      "value": "%"
         write 0;
                                                                                   }
                                                                                  }
                                                                                 "identifier": "s",
                                                                                "type": ":="
```

```
Vite App
                                                                                                                              保存输出结果到文件
                                                                    输出
     输入
                                                                                          "single-expression": {
                                                                                           "expression": [
       read x;
                                                                                            {
       read y;
                                                                                              "op": {
       for i := x downto 1 do
                                                                                                "leftExpression": [
         s := x % y;
                                                                                                {
                                                                                                 "identifier": "x"
          s += 1;
                                                                                                }
          s := s and 3;
                                                                                                ],
          for j := 1 to y do
                                                                                                "rightExpression": [
            m := 1 + (2*_X);
                                                                                                {
             if (m <> 0)
                                                                                                 "identifier": "y"
                                                                                               }
              write m;
             else
                                                                                               "value": "+"
             write 0;
          repeat
                                                                                            }
           fact := fact * x;
           x := x - 1;
                                                                                           "op": "not"
          until x = 0;
                                                                                        }
          enddo;
       enddo;
                                                                                      "rightExpression": [
                                                                                      {
    "const": 10
  }
       if (not (x + y)*10 >= 100)
        write 100;
                                                                                      ],
       else
                                                                                      "value": "*"
          write 0;
                                                                                    }
                                                                                  "rightExpression": [
                                                                                  {
  "const": 100
```

测试数据三



总结

可以正确的识别单元运算符 (not) 和 += , -= 等扩充的运算符号。也可以正确的识别 for 循环和扩充的比较运算符号 (>= , <> , <=) , 也支持位运算符。但是这里识别正则表达式有些问题,如果遇到正则表达式会报错。