==<font color="#800000">'''关键词'''</font>==

<font color="#808080">Assembler、Methematica、demo、Fuction、symboltable、import、Split.

</font>

==<font color="#800000">'''逻辑模型'''</font>==

<font color="#808080">

[[File:逻辑模型07wsj.png|1100px]]

</font>

==<font color="#800000">'''学习报告正文'''</font>==

===<font color="#800000">'''课程目标'''</font>===

<font color="#808080">

\*完成project6

\*完成第七周学习报告，总结第六周学习内容

\*运用Mathematica完成第六周的作业

</font>

===<font color="#800000">'''课程内容'''</font>===

<font color="#808080">

====<font color="black">'''1.Assembler（MMA版）'''</font>====

\*和其他版的Assembler一样，分为Parser，Code，Symboltable部分。

\*\*<font color="black">'''Parser'''</font>

\*\*\*<font color="black">'''调用函数:'''</font>

\*\*\*1.Import,Export,If,While,++,"",Length等基本函数，在这里不再赘述。

\*\*\*2.StringSplit[[File:StringSplit.png|400px]][[File:SringSplit demo.png|400px]]

\*\*\*通过字符“\n”（换行）将导入的文本转化为很多行，每一行都作为一个元素存到一个全局变量“test”中（Parser的思路完全借鉴顾老师）

\*\*\*3.StringMatchQ[[File:StringMatchQ.png|600px]]

\*\*\*这个函数有一些坑，它的匹配一种模式是完全匹配，而不是从一段字符串中判断是否含有某个字符串，但这样恰恰增加了更多模式的可能，所以如果pattern只输入字符串，那么前后字符串要完全相同才会返回True，例如

\*\*\*[[File:StringMatchQ demo2.png]][[File:StringMatchQ demo3.png]]

\*\*\*除此之外，这个函数还有一些特殊的地方，就是第一个图中说的，它的pattern中如果包含“@”“\*”等元字符，其代表意义会发生列表中的那种改变，但是通过Verbatim可以取消他们的特殊含义。

\*\*\*3.1Verbatim[[File:Verbatim.png|400px]]

\*\*\*3.2StringExpression[[File:StringExpression.png|400px]]

\*\*\*这里注意到使用StringExpression中的\_\_\_来创建一种模式“//”~~\_\_\_时其指代的意思变成了//后接零个或多个字符这样一种模式，所以我们就能够通过这几个函数的组合来判定A指令行了[[File:StringMatchQ demo1.png]]

\*\*\*4.StringTrim[[File:StringTrim.png|400px]]

\*\*\*可以用这个函数去除多余的空白字符，也可以把A指令前的@删去。

\*\*\*5.StringExtract[[File:StringExtract.png|400px]]

\*\*\*可以用这个函数来分别提取“//”前，“=”左右，“；”左右的字符例如

\*\*\*[[File:StringExtract demo.png]]

\*\*\*6.AppendTo（把某一元素加入到某集合的最尾端）[[File:AppendTo.png|400px]]

\*\*\*7.Module(定义局部变量)[[File:Module.png|400px]]

\*\*\*8.Return(用于循环完之后输出得到的Hack代码和机器码等)[[File:Return.png|400px]]

\*\*\*'''基本思路'''

\*\*\*1.用Module定义局部变量并赋予初始值“empryArry={}（相当于空集）”，“n=1”

\*\*\*2.用StringSplit以‘\n’为分隔符把文档划分为很多行，并用StringTrim把多余空白删去，最后存到一个集合Array里。

\*\*\*3.做一个循环，当n<=Array长度时，执行以下操作，其中的自变量是Array[[n]]。

\*\*\*4.用StringMatchQ判断是否为注释行，以及用If函数来判断是否为空行（字符串长度为0）

\*\*\*5.用IF函数，如果一行既不是空行也不是注释行，就用StringExtract来提取“//”前的文本,这样做是考虑到可能有注释行和代码在一行然后把获得的字符串删去前后空白，这样得到的代码就前后都没有空白了，最后用AppendTo把这一行存到emptyArray中。

\*\*\*6.用Return导出emptyArray中的元素。

\*\*<font color="black">'''Code'''</font>

\*\*\*<font color="black">'''调用函数:'''</font>

\*\*\*Rule[[File:Rule.png|400px]]顾老师已经向我们展示过了

\*\*\*我的部分Rule[[File:Rules.png|400px]]

\*\*\*StringJoin[[File:StringJoin.png|500px]]

\*\*\*用于连接字符串，在这里我用来把分散的dest，comp等指令部分的二进制数连接成完整的16位指令的二维码，例如

\*\*\*[[File:SomeCode.png|800px]]

\*\*\*'''基本思路：'''

\*\*\*判断一行是否为C指令，如果为C指令，则用StringExtract来提取“=”前后的字符，分别用一定Rule来对应

\*\*\*最后用字符串连接符把所有部分的字符连接起来，组成一条完整的C指令翻译成的机器码。

\*\*<font color="black">'''Symboltable'''</font>

\*\*\*<font color="black">'''调用函数:'''</font>

\*\*\*IntegerString[[File:IntegerString.png|400px]]

\*\*\*非常好用的一个函数，省去了其他语言中自己编写类似函数的麻烦

\*\*\*LeeterCharacter[[File:LeeterCharacter.png|400px]]

\*\*\*用来判断@后是否跟有字母，但是有些坑

\*\*\*[[File:字母demo.png|400px]]

\*\*\*'''基本思路：'''

\*\*\*1.判定是A指令还是C指令还是标签。

\*\*\*2.在A指令判定为True之后，判断是否有字母，如果有，则判断是否已存在，如果不存在，则创建一个新的变量名存到一个特定的存储字符的集合中。

\*\*\*3.将@后的值用IntegerString转化为16位数值存为机器码的一行.

\*\*\*4.C指令，标签以及C指令中的Jump类似的可以通过所有已给的函数中的几个处理成机器码。

====<font color="black">'''2.顾老师的补充以及对课程的思考感悟'''</font>====

\*上节课最后走的时候和顾老师聊到了很晚，对原先对计算思维和系统设计有了一些新的理解

\*\*最开始我以为计算思维就是对多层函数的构建，从而设计出一个庞大复杂的系统，这个系统像计算机一样分为多个抽象层，来处理各种问题。

\*\*现在计算思维恰恰是我们如何去构建所需的多层函数，也就是一种策略，然后如何对这种策略进行评估。

\*\*在2006年3月由美国卡内基·梅隆大学计算机科学系主任周以真（Jeannette M. Wing）教授在美国计算机权威期刊《Communications of the ACM》杂志上给出它的一种定义：计算思维是运用计算机科学的基础概念进行问题求解、系统设计、以及人类行为理解等涵盖计算机科学之广度的一系列思维活动。[http://baike.baidu.com/link?url=qwI8QOS4AnKy1YmlRggLftQlb2NzOilf1-nzrU0Yoe8zhbNziCXa-2GIJBi\_duVs4R2zJPghL0hlqx8PSbg8VpyyGuDsJadBzvtPr-uFCi0yBosBgcIPcjSEERdGPwAW]

\*\*它分为三个过程[https://en.wikipedia.org/wiki/Computational\_thinking], 1) Problem Formulation (abstraction),也就是对问题的构建以及抽象化，2) Solution Expression (automation)自动化的解决方案3) Solution Execution & Evaluation (analyses)解决方案的执行和评估

\*\*[[File:Computational thinking.jpg|400px]]

\*\* Currently Computational Thinking is broadly defined as a set of cognitive skills and problem solving processes that include (but are not limited to) the following characteristics[1][2]:

\*\*\* Using abstractions and pattern recognition to represent the problem in new and different ways

\*\*\* Logically organizing and analyzing data

\*\*\* Breaking the problem down into smaller parts

\*\*\* Approaching the problem using programmatic thinking techniques such as iteration, symbolic representation, and logical operations

\*\*\* Reformulating the problem in order for it to be solved using a series of ordered steps (algorithmic thinking)

\*\*\* Identifying, analyzing, and implementing possible solutions with the goal of achieving the most efficient and effective combination of steps and resources

\*\*\* Generalizing and transferring this problem solving process to a wide variety of problems

====<font color="black">'''3.Mathematica新范例的尝试'''</font>====

'''1.用MMA做微积分作业（手动滑稽）'''

\*[[File:一道极限题.jpg|600px]]

\*[[File:MMA解答.png|600px]]

'''2.月亮以太阳为参照物的运动轨迹的构建'''

在学习《经度》时提到了月亮绕太阳转是一个类似多边形的东西于是想用MMA探究一下

\*首先要知道地球到太阳和地球到月亮的距离

\*[[File:Distance.png|400px]]

\*[[File:黑人问号.png|400px]]au是啥？？？

\*[[File:au.png|600px]]

\*[[File:单位换算.png|500px]]

\*原来au是天文单位，它是一个长度单位，长度已给出。

\*通过两个参数方程确定其运动曲线

\*\*[[File:plot.png|600px]]

\*\*1.首先定义月球绕地球转的半径为单位长度，地球绕太阳转的角速度为单位角速度，那么p1变量就是地球绕太阳的半径，p2变量是月球绕地球转的角速度，因为地球在绕太阳转，月球在绕地球转，所以会产生两个向量，一个确定地球的坐标（p1\*Sin[t] ，p1\*Cos[t]），另一个确定因为月球的转动使月球位置发生偏移的向量（Sin[p2\*t]，Cos[p2\*t]）两向量相加即为月球的位置。

\*\*2.然后输出图像

\*\*[[File:dmsd.png|500px]]

\*\*[[File:dms.gif|600px]]

\*\*可以看到它的图像确实有些像多边形

\*\*3.分析它的曲率半径

\*\*[[File:曲率半径.png|1000px]]

\*\*[[File:曲率半径图.png|800px]]

\*\*可以看到其曲率半径恒为正，也就是恒为外凸的。

\*\*放上p2从0到366变化中的几张图

\*\*[[File:M1.png|400px]][[File:M2.png|400px]][[File:M3.png|400px]][[File:M4.png|400px]]

</font>

==<font color="#800000">'''参考文献'''</font>==

[1]Grover, Shuchi; Pea, Roy (2013). "Computational Thinking in K–12 A Review of the State of the Field". Educational Researcher. 42.

[2]Stephenson, Chris; Valerie Barr (May 2011). "Defining Computational Thinking for K-12". CSTA Voice. 7 (2): 3–4. ISSN 1555-2128. CT is a problem solving process...

Mathematica帮助文件

[http://nand2tetris.org/02.php| Nand2Tetris project6]

[https://en.wikipedia.org/wiki/Computational\_thinking|Computational thinking Wikipedia]

==版权声明==

[[File:BYNCND.png | link=http://creativecommons.org/licenses/by-nc-nd/3.0/cn/ ]] [http://creativecommons.org/licenses/by-nc-nd/3.0/cn/ 署名-非商业性使用-禁止演绎 3.0 中国大陆 (CC BY-NC-ND 3.0 CN)]