编译原理

实

验

报

告

**实验名称：实验一：词法分析**

**姓名： 方澳阳**

**学号： 180110115**

**学院： 计算机科学与技术**

**专业： 计算机类**

# 实验目的与方法

（1）通过本实验加深对词法分析程序的功能及实现方法的理解

（2） 设计并编程实现一个词法分析程序，对C语言源程序段进行词法分析，加深对高级语言的认识

（3）对C语言的文法描述有更深的认识，体会有穷自动机，编码表和符号表在编译的整个过程中的应用

# 实验总体流程与函数功能描述

**流程：**

1. 将代码串读取进来
2. 根据编码表初始化c实验的种别码
3. 对读进来的代码串进行扫描，循环判断读进来的字符串，将其转为token。
4. 遍历所有token串，将标识符存到符号表中，并且初始化其他字段。

read\_program(): 从txt文件中读取代码,存到program

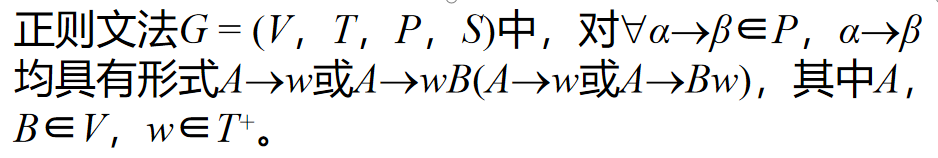
initMap(): 初始化C语言的种别码

getNext(); 使pos指针加1,使ch为下一个字符

scanner(); 扫描器,对ch进行判断,存入对应的token串中

# 实验内容

使用正则文法:



**标识符\关键字的文法描述：**

****

****

****

**常数的文法描述：**

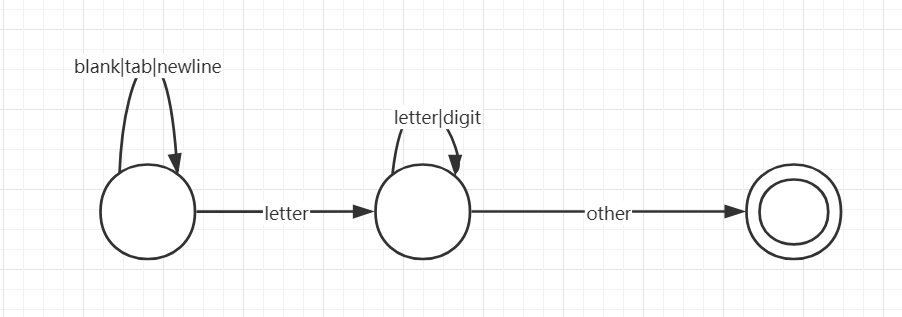
****

**运算符的文法描述:**

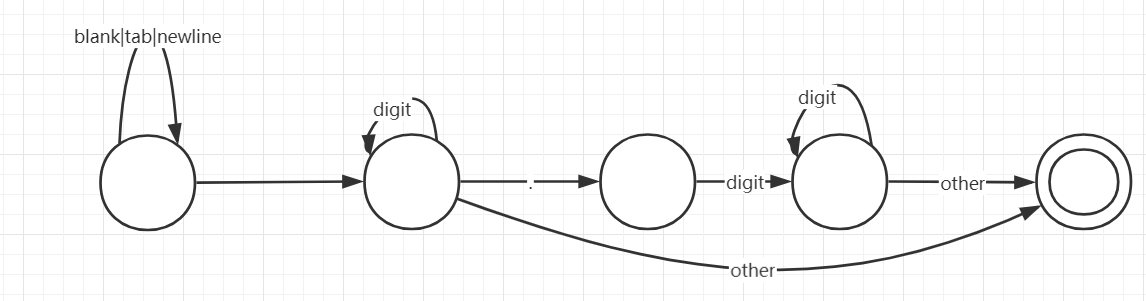
****

**有穷自动机:**

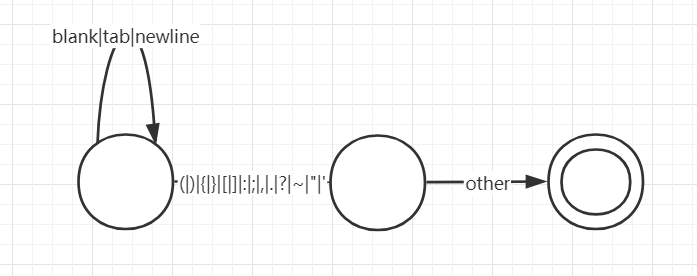
1. 检测标识符\关键字



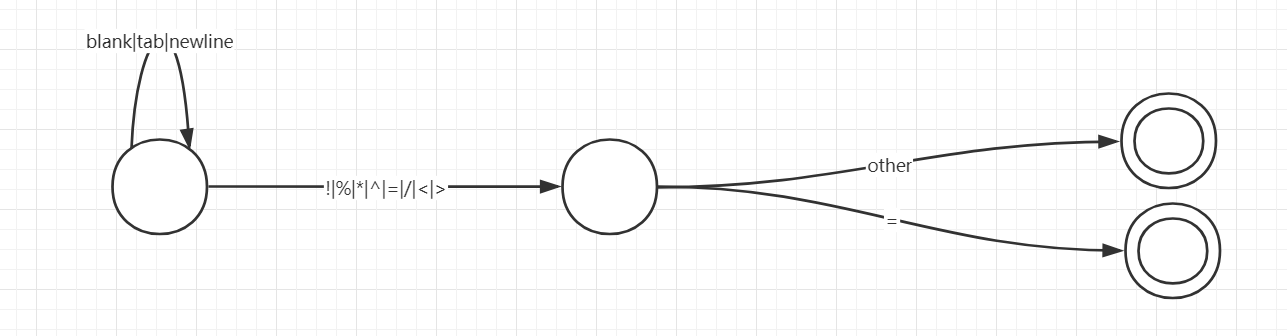
1. 检测常数

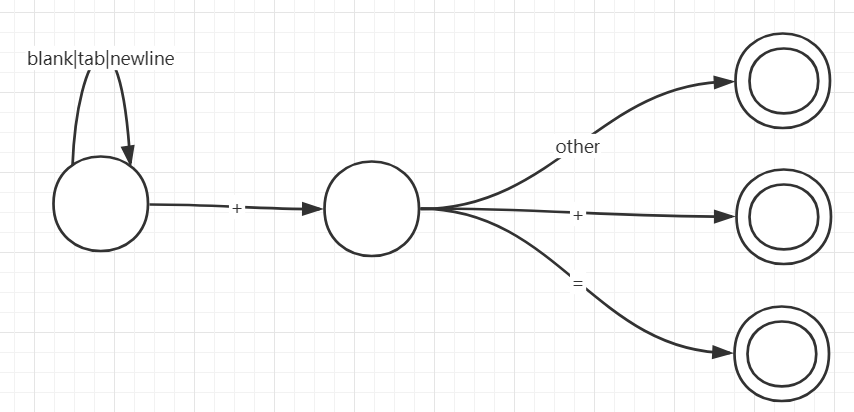


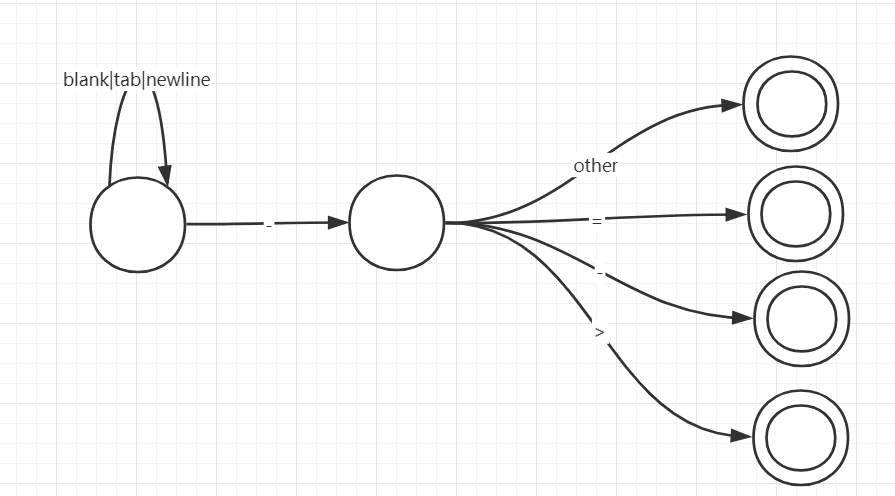
1. 检测分界符以及特殊运算符(因为他们的状态转移相同, 所以合并成一个状态)

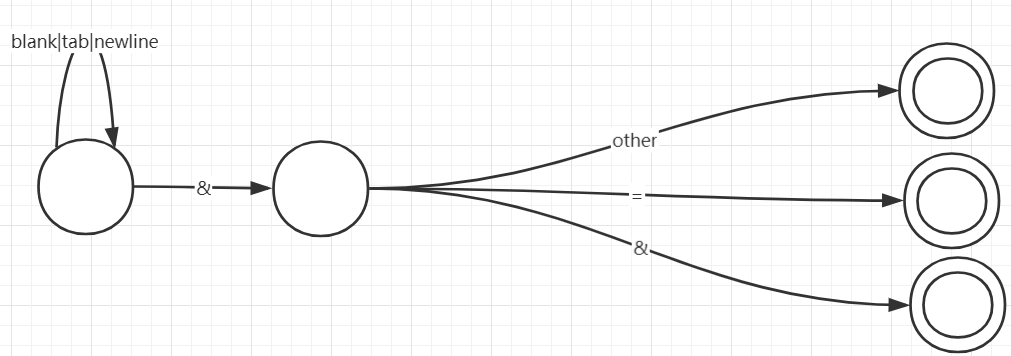


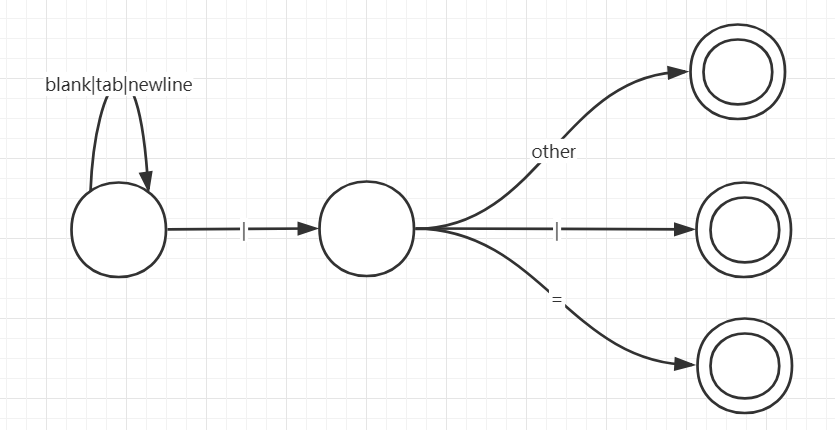
1. 检测运算符











**编码表：**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| auto | 1 | switch | 26 | / = | 51 |
| break | 2 | typedef | 27 | : | 52 |
| case | 3 | union | 28 | ; | 53 |
| char | 4 | unsigned | 29 | ? | 54 |
| const | 5 | void | 30 | [ | 55 |
| continue | 6 | volatile | 31 | ] | 56 |
| default | 7 | while | 32 | ^ | 57 |
| do | 8 | - | 33 | ^ | 58 |
| double | 9 | = | 34 | { | 59 |
| else | 10 | -= | 35 | | | 60 |
| enum | 11 | -> | 36 | || | 61 |
| extern | 12 | ! | 37 | |= | 62 |
| float | 13 | != | 38 | } | 63 |
| for | 14 | % | 39 | ~ | 64 |
| goto | 15 | %= | 40 | + | 65 |
| if | 16 | & | 41 | ++ | 66 |
| int | 17 | && | 42 | += | 67 |
| long | 18 | &= | 43 | < | 68 |
| register | 19 | ( | 44 | << | 69 |
| return | 20 | ) | 45 | <<= | 70 |
| short | 21 | \* | 46 | <= | 71 |
| signed | 22 | \* | 47 | = | 72 |
| sizeof | 23 | , | 48 | == | 73 |
| static | 24 | . | 49 | > | 74 |
| struct | 25 | / | 50 | >= | 75 |
| >> | 76 | \ | 78 | # | 80 |
| >>= | 77 | \= | 79 | /\*注释\*/ | 81 |
| 常数 | 82 | 标识符 | 83 |  |  |

## Token串和符号表的逻辑结构及存贮结构

Token串的存储使用一个动态数组存储，定义为

vector<pair<int,string>>

数组中的每一个元素为一个pair，pair.first代表种类，pair.second表示具体的串。当pair.first为标识符时，可以通过pair.second去符号表中索引到对应的信息。

符号表使用Map作为存储结构。通过设置对应的string串作为key，可以在Map中得到其对应的value。符号表定义为

map<string,SYSTABLE> sysMap

其中，SYSTABLE为一个类，里面存储了对应标识符的type和address。

## 算法描述

1. 由于在实现过程中需要大量地用到“将指针移动到下一个字符”的操作,因此将该操作封装成一个函数，并且包括错误检测。

bool getNext(char &ch, string::size\_type &pos, const string &program) {

pos++;

if (pos >= program.size()) {

return false;

} else {

ch = program[pos];

return true;

}

}

1. 判断用到了sanner函数，其逻辑为:
   1. 先判断从指针pos开始的字符串是否为标识符或者关键字，若是，则按照正则文法给出的状态机将pos往后移，直到遇到不符合标识符或关键字文法的字符。
   2. 再判断是否为常数。
   3. 再判断是否为标识符。
   4. 函数返回一个pair，表示一个token。
2. 以上两步完成后, 则需要遍历所有的token串。
   1. 如果token串表明该串是一个标识符，则将该串作为key，新建一个value对象（即SYSTABLE），填入到符号表中。
   2. 如果不是则跳过
   3. 重复以上步骤，直到遍历结束所有的token

由于C++中的map可以保证关键字的唯一性，所以该方法可行。

# 实验结果与分析

**输入的代码串,该代码实现的是冒泡排序:**

#include <stdio.h>

void bubble\_sort(int a[], int n);

int number[10000000];

void bubble\_sort(int a[], int n)

{

int i,j,temp;

for (j=0;j<n-1;j++)

{

for (i=0;i<n-1-j;i++)

{

if(a[i]>a[i+1])

{

temp=a[i];

a[i]=a[i+1];

a[i+1]=temp;

}

}

}

}

int main()

{

int i,n;

scanf("%d",&n);

for(int j=0;j<n;j++)

scanf("%d",&number[j]) ;

bubble\_sort(number,n);

for (i=0;i<n-1;i++)

printf("%d ",number[i]);

printf("%d\n",number[i]);

return 0;

}

## 输出的token串:

(80,#)

(83,include)

(68,<)

(83,stdio)

(82,.)

(83,h)

(74,>)

(30,void)

(83,bubble\_sort)

(44,()

(17,int)

(83,a)

(55,[)

(0,])

(48,,)

(17,int)

(83,n)

(45,))

(53,;)

(17,int)

(83,number)

(55,[)

(82,10000000)

(0,])

(53,;)

(30,void)

(83,bubble\_sort)

(44,()

(17,int)

(83,a)

(55,[)

(0,])

(48,,)

(17,int)

(83,n)

(45,))

(59,{)

(17,int)

(83,i)

(48,,)

(83,j)

(48,,)

(83,temp)

(53,;)

(14,for)

(44,()

(83,j)

(72,=)

(82,0)

(53,;)

(83,j)

(68,<)

(83,n)

(33,-)

(82,1)

(53,;)

(83,j)

(66,++)

(45,))

(59,{)

(14,for)

(44,()

(83,i)

(72,=)

(82,0)

(53,;)

(83,i)

(68,<)

(83,n)

(33,-)

(82,1)

(33,-)

(83,j)

(53,;)

(83,i)

(66,++)

(45,))

(59,{)

(16,if)

(44,()

(83,a)

(55,[)

(83,i)

(0,])

(74,>)

(83,a)

(55,[)

(83,i)

(65,+)

(82,1)

(0,])

(45,))

(59,{)

(83,temp)

(72,=)

(83,a)

(55,[)

(83,i)

(0,])

(53,;)

(83,a)

(55,[)

(83,i)

(0,])

(72,=)

(83,a)

(55,[)

(83,i)

(65,+)

(82,1)

(0,])

(53,;)

(83,a)

(55,[)

(83,i)

(65,+)

(82,1)

(0,])

(72,=)

(83,temp)

(53,;)

(63,})

(63,})

(63,})

(63,})

(17,int)

(83,main)

(44,()

(45,))

(59,{)

(17,int)

(83,i)

(48,,)

(83,n)

(53,;)

(83,scanf)

(44,()

(78,")

(39,%)

(83,d)

(78,")

(48,,)

(41,&)

(83,n)

(45,))

(53,;)

(14,for)

(44,()

(17,int)

(83,j)

(72,=)

(82,0)

(53,;)

(83,j)

(68,<)

(83,n)

(53,;)

(83,j)

(66,++)

(45,))

(83,scanf)

(44,()

(78,")

(39,%)

(83,d)

(78,")

(48,,)

(41,&)

(83,number)

(55,[)

(83,j)

(0,])

(45,))

(53,;)

(83,bubble\_sort)

(44,()

(83,number)

(48,,)

(83,n)

(45,))

(53,;)

(14,for)

(44,()

(83,i)

(72,=)

(82,0)

(53,;)

(83,i)

(68,<)

(83,n)

(33,-)

(82,1)

(53,;)

(83,i)

(66,++)

(45,))

(83,printf)

(44,()

(78,")

(39,%)

(83,d)

(78,")

(48,,)

(83,number)

(55,[)

(83,i)

(0,])

(45,))

(53,;)

(83,printf)

(44,()

(78,")

(39,%)

(83,d)

(0,\n)

(78,")

(48,,)

(83,number)

(55,[)

(83,i)

(0,])

(45,))

(53,;)

(20,return)

(82,0)

(53,;)

(63,})

## 输出的符号表

(a, -1, ?)

(bubble\_sort, -1, ?)

(d, -1, ?)

(h, -1, ?)

(i, -1, ?)

(include, -1, ?)

(j, -1, ?)

(main, -1, ?)

(n, -1, ?)

(number, -1, ?)

(printf, -1, ?)

(scanf, -1, ?)

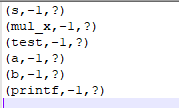
(stdio, -1, ?)

(temp, -1, ?)

# 实验中遇到的困难与解决办法

在实验的过程中一开始不知道要做什么，因为从理论到实际还是有一定差距的。并且在实验指导书中给出的输出的例子有一些错误，指导思想不是很明确。再加上自己对于编译的整个过程不是很理解，导致不明白一些概念。

例如符号表中后面这些-1和？的含义。



在与老师沟通交流之后，才明白这些空白需要在后面语法分析、语法制导等过程中再逐渐补充的。并且也知道了实现方式可以多种多样，例如我这次使用了c++的一些STL库中的数据结构，这比自己用c造出一个数据结构更加方便。

但是这次实验做的也有很多不足，例如使用了大量的switch case语句，导致代码重复性高，没有尽可能地复用代码。整个词法分析的实现也相当的“原始”，只是对有可能出现的情况进行分类讨论，没有做到根据输入动态调整。